Package ‘warbleR’

November 4, 2019

Type Package
Title Streamline Bioacoustic Analysis
Version 1.1.18
Date 2019-10-29
Description Functions aiming 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 basic sound analysis tools from the package ‘seewave’, 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, RCurl, rjson, stats, utils, methods, pracma, Sim.DiffProc, maps
Depends R (>= 3.2.1), tuneR, seewave (>= 2.0.1), NatureSounds
LazyData TRUE
BugReports https://github.com/maRce10/warbleR/issues
NeedsCompilation no
Suggests knitr, ggplot2, rmarkdown, jpeg, ape, bioacoustics, soundgen, wavethresh
VignetteBuilder knitr
RoxygenNote 6.1.1
Repository CRAN
Language  en-US
Date/Publication  2019-11-04 06:20:03 UTC
Author  Marcelo Araya-Salas [aut, cre]
        (<https://orcid.org/0000-0003-3594-619X>),
        Grace Smith-Vidaurre [aut]
Maintainer  Marcelo Araya-Salas <marceloa27@gmail.com>

R topics documented:

autodetec ......................................................... 3
catalog ............................................................ 7
catalog2pdf ...................................................... 12
checksels ......................................................... 13
checkwavs ......................................................... 15
color.spectro ..................................................... 17
compare.methods .................................................. 19
consolidate ....................................................... 23
coor.graph ......................................................... 25
coor.test ........................................................... 27
cut_sels ............................................................ 29
dfDTW ................................................................. 31
dfts ................................................................. 34
ffDTW ................................................................. 36
ffts ................................................................. 39
filtersels ............................................................ 41
fixwavs .............................................................. 43
fix_extended_selection_table .................................... 44
frange ............................................................... 45
frange.detec ....................................................... 48
inflections ......................................................... 50
is_extended_selection_table ..................................... 52
is_selection_table ............................................... 53
lbh_selec_table ................................................... 54
lspec ............................................................... 55
lspec2pdf .......................................................... 57
manualoc ............................................................ 59
mfcc_stats ......................................................... 61
move.imgs .......................................................... 63
mp32wav ............................................................ 65
multi_DTW .......................................................... 66
new_function_names .............................................. 68
open_wd ............................................................. 69
ovlp_sels .......................................................... 70
phylo_spectro ..................................................... 71
querxc .............................................................. 74
read_wave .......................................................... 76
resample_est ....................................................... 78
**autodetect**

*Automatically detect vocalizations in sound files*

**Description**

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

**Usage**

```r
autodetect(X = NULL, threshold = 15, envt = "abs", ssmooth = NULL, msmooth = NULL,
  power = 1, bp = NULL, osci = FALSE, wl = 512, xl = 1, picsize = 1, res = 100,
  flim = c(0, 22), ls = FALSE, sxrow = 10, rows = 10, mindur = NULL, maxdur =
  NULL, redo = FALSE, img = TRUE, it = "jpeg", set = FALSE, flist = NULL, smadj = NULL,
  parallel = 1, path = NULL, pb = TRUE, pal = reverse.gray.colors.2,
  fast.spec = FALSE, ...)```

Arguments

`X` 'selection_table' object or a data frame with columns for sound file name (sound.files), selection number (select), and start and end time of signal (start and end). If provided the detection will be conducted only within the selections in 'X'.

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

`envt` Character vector of length 1 specifying the type of envelope to be used: "abs" for absolute amplitude envelope or "hil" for Hilbert amplitude envelope. Default is "abs".

`ssmooth` A numeric vector of length 1 to smooth the amplitude envelope with a sum smooth function. Default is NULL.

`msmooth` A numeric vector of length 2 to smooth the amplitude envelope with a mean sliding window. The first component is the window length and the second is the overlap between successive windows (in %). Faster than ssmooth but time detection is much less accurate. Will be deprecated in future versions. Default is NULL.

`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).

`bp` Numeric vector of length 2 giving the lower and upper limits of a frequency bandpass filter (in kHz). Default is c(0, 22).

`osci` Logical argument to add an oscillogram underneath spectrogram, as in `spectro`. Default is FALSE. Not applied if `ls` is TRUE.

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

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

`picsize` Numeric argument of length 1. Controls the relative size of the spectrogram. Default is 1.

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

`flim` A numeric vector of length 2 for the frequency limit in kHz of the spectrogram, as in `spectro`. Default is c(0, 22).

`ls` Logical argument. If TRUE, long spectrograms as in lspec are produced.

`sxrow` A numeric vector of length 1. Specifies seconds of spectrogram per row when creating long spectrograms. Default is 10. Applied when `ls` = TRUE and/or when `X` is not provided.

`rows` A numeric vector of length 1. Specifies number of rows per image file when creating long spectrograms. Default is 10. Applied when `ls` = TRUE and/or when `X` is not provided.

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

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

redo Logical argument. If TRUE all selections will be analyzed again when code is rerun. If FALSE only the selections that do not have an 'autodetect' generated image file in the working directory will be analyzed. Default is FALSE.

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".

set A logical argument indicating whether the settings of the autodetection process should be included in the image file name. If TRUE, threshold (th), envelope (envt), bandpass (bp), power (pw), smooth (smo, either msmooth[1] or ssmooth), maxdur (mxdu), and mindur (midu) are included.

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

smadj adjustment for amplitude smoothing. Character vector of length one indicating whether start end values should be adjusted. "start", "end" or "both" are the inputs admitted by this argument. Amplitude smoothing through ssmooth generates a predictable deviation from the actual start and end positions of the signals, determined by the threshold and smooth values. This deviation is more obvious (and problematic) when the increase and decrease in amplitude at the start and end of the signal (respectively) is not gradual. Ignored if ssmooth is NULL.

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.

cal 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.

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.

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")`. The output of `manualoc` can be used as the input data frame. This function uses a modified version of the `timer` function from seewave package to detect signals.

**Value**

Image files with spectrograms showing the start and end of the detected signals. It also returns a data frame containing the start and end of each signal by sound file and selection number.

**Author(s)**

Marcelo Araya-Salas (<marceloa27@gmail.com>). Implements a modified version of the timer function from seewave.

**References**


**Examples**

```r
## Not run:
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))
writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav"))
writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))
ad <- autodetec(threshold = 5, env = "hil", ssmooth = 300, power=1,
bp=c(2,9), xl = 2, picsize = 2, res = 200, flim= c(1,11), osci = TRUE,
wl = 300, ls = FALSE, sxrow = 2, rows = 4, mindur = 0.1, maxdur = 1, set = TRUE, path = tempdir())

#run it with different settings
ad <- autodetec(threshold = 90, env = "abs", ssmooth = 300, power = 1, redo = TRUE,
bp=c(2,9), xl = 2, picsize = 2, res = 200, flim= c(1,11), osci = TRUE,
wl = 300, ls = FALSE, sxrow = 2, rows = 4, mindur=0.1, maxdur=1, set = TRUE, path = tempdir())

#check this folder!!
tempdir()
```

```r
## End(Not run)
```
Create catalog of vocal signals

Description

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

Usage

catalog(X, flim = c(0, 22), 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, wl = 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 spectro. Default is c(0,22).

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 spec-
    trogram. 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 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 num-
    ber 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 ftwindow for name options. Default is "hanning". |
| **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. Palettes as gray.2 may work better when 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. |
| **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. |
| **res** | 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. |
| **orientation** | 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. |
| **labels** | String vector. Provides the column names that will be used as labels above the corresponding spectrograms. |
| **height** | Numeric. Single value (in inches) indicating the height of the output image files. Default is 11 for vertical orientation. |
| **width** | Numeric. Single value (in inches) indicating the width of the output image files. Default is 8.5 for vertical orientation. |
| **tags** | 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. |
| **tag.pal** | List of color palette function for tags. Should be of length 1, 2 or 3. Default is list(temp.colors,heat.colors,topo.colors). |
**legend**

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.

**cex**

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

**leg.wd**

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

**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 (no suffix). Useful to label catalogs from different individuals, species or sites.

**img.prefix**

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.

**tag.widths**

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. are

Details

This functions 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).

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 (<marceloa27@gmail.com>)

References


See Also

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

Examples

## 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, wl = 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, wl = 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, wl = 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())

check this floader
tempdir()
## End(Not run)

catalog2pdf

catalog2pdf combines catalog images into pdfs

### Description

catalog2pdf combines catalog images into pdfs

### Usage

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. (including sound file and page number) should be magnified. Default is TRUE.
- **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. 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 (<marceloa27@gmail.com>)

### References

checksels

See Also
catalog2pdf, 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 folder

## End(Not run)

----

checksels  

Check selection data frames

Description

checksels checks whether selections can be read by subsequent functions.

Usage

checksels(X, parallel = 1, path = NULL, check.header = FALSE, pb = TRUE, wav.size = FALSE)

Arguments

X  
'selection_table' object or data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. Alternatively, a 'selection_table' class object can be input to double check selections. The output of manualoc or autodetec can be used as the input data frame.

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.

check.header  
Logical. Controls whether sound file headers correspond to the actual file properties (i.e. if is corrupted). This could significantly affect the performance of the function (much slower) particularly with long sound files.
Logical argument to control progress bar. Default is TRUE.

Logical argument to control if the size of the wave object when the selection is imported into R (as when using `readWave` is calculated and added as a column. Size is return in MB. Default is FALSE.

Details

This function checks the information in a selection data frame or selection table (i.e. data frame with annotations on sound files) to avoid problems in any warbleR analysis downstream. It specifically checks if:

- 'X' is an object of class 'data.frame' or 'selection_table' (see `selection_table`) and contains the required columns to be used on any warbleR function ('sound.files', 'selec', 'start', 'end', if not returns an error)
- 'sound.files' in 'X' correspond to .wav files in the working directory or in the provided 'path' (if no file is found returns an error, if some files are not found returns error info in the output data frame)
- time ('start', 'end') and frequency ('bottom.freq', 'top.freq', if provided) limit parameters are numeric and don’t contain NAs (if not returns an error)
- there are no duplicated selection labels ('selec') within a sound file (if not returns an error)
- sound files can be read (error info in the output data frame)
- the start and end time of the selections are found within the duration of the sound files (error info in the output data frame)
- sound files can be read (error info in the output data frame)
- sound files header is not corrupted (only if `header = TRUE`, error info in the output data frame)
- selection time position (start and end) doesn’t exceeds sound file length (error info in the output data frame)
- 'top.freq' is lower than half the sample rate (nyquist frequency, error info in the output data frame)
- negative values aren’t found in time or frequency limit parameters (error info in the output data frame)
- 'start' higher than 'end' or 'bottom.freq' higher than 'top.freq' (error info in the output data frame)
- 'channel' value is not higher than number of channels in sound files (error info in the output data frame)

The function returns a data frame that includes the information in 'X' plus additional columns about the format of sound files (see 'Value') as well as the result of the checks ('check.res' column, value is 'OK' if everything is fine). Sound files should be in the working directory (or the directory provided in 'path'). Corrupt files can be fixed using `fixwavs`.

Value

A data frame including the columns in the input data frame (X) and the following additional columns:
checkwavs  

- check.res: diagnose for each selection  
- duration: duration of selection in seconds  
- min.n.samples number of samples in a selection. Note the number of samples available in  
  a selection limits the minimum window length (wl argument in other functions) that can be  
  used in batch analyses.  
- sample.rate: sampling rate in kHz  
- channels: number of channels  
- bits: bit depth  
- sound.file.samples: number of samples in the sound file

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References

Araya-Salas, M., & Smith-Vidaurre, G. (2017). warbleR: An R package to streamline analysis of  

See Also

checkwavs

Examples

{
  # 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"))

  checksels(X = lbh.selec_table, path = tempdir())
}

checkwavs  

Check .wav files

Description

checkwavs checks whether .wav files can be read by subsequent functions.

Usage

checkwavs(X = NULL, path = NULL)
Arguments

X  Optional. ‘selection_table’ object or data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of manualoc or autodetect can also be used as the input data frame. 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. specan).

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 .wav files in the working directory can be read. Users must set the working directory where they wish to check .wav 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 fixwavs ('sox' must be installed to be able to run this function). The function is intended for a "quick and dirty" check of the .wav files in a selections data frame. For a more thorough analysis see checksels.

Value

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

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

checksels seltailor

Examples

{
  # save wav 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"))

  # without selection data frame

}
checkwavs(path = tempdir())

# without selection data frame
checkwavs(X = lbh_selec_table, path = tempdir())
}

---

**color.spectro**  
*Highlight spectrogram regions*

**Description**

`color.spectro` highlights spectrogram regions specified by users.

**Usage**

```r
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", 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.cm' 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'.
**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 (<marceloa27@gmail.com>) and Grace Smith Vidaurre
**compare.methods**

**Assessing the performance of acoustic distance measurements**

**Description**

`compare.methods` creates graphs to visually assess performance of acoustic distance measurements.

**References**


**See Also**

- trackfreqs for creating spectrograms to visualize frequency measurements by specan, snrspecs
- for creating spectrograms to optimize noise margins used in sig2noise

Other spectrogram creators: dfDTW, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, speccreator, trackfreqs

**Examples**

```r
## 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(-90, 0, 5),
              dB = "B", X = st, col.clm = "colors", base.col = "skyblue", t.mar = 0.07, f.mar = 0.1,
              interactive = NULL)

# interactive (selected manually: you have to select them by clicking on the spectrogram)
color.spectro(wave = sgnl, wl = 300, ovlp = 90, flim = c(1, 8.6), collevels = seq(-90, 0, 5),
              dB = "B", col.clm = "colors", t.mar = 0.07, f.mar = 1, interactive = 2)

## End(Not run)
```
Usage

```r
compare.methods(X = NULL, flim = c(0, 22), bp = c(0, 22), 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 `manualoc` function, or any data frame with columns for sound file name (sound.files), selection number (select), 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 c(0, 22).
- **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 c(0, 22).
- **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 (`xcorr` function)
  - **dfDTW**: dynamic time warping on dominant frequency contours (`dfDTW` function)
  - **ffDTW**: dynamic time warping on fundamental frequency contours (`ffDTW` function)
  - **SP**: spectral parameters (`specan` function)
  - **SPharm**: spectral parameters (`specan` function with argument `harmonicity = TRUE`)
  - **MFCC**: statistical descriptors of Mel frequency cepstral coefficients (`mfcc_stats` function)
The `compare.methods` 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.

**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**
TO BE DEPRECATED. Use "custom1" and "custom2" arguments instead.

**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.

**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 file 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 `specreator`).

**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 xcorr), dynamic time warping on dominant frequency time series (dfDTW, from dtw applied on dfts output), dynamic time warping on dominant frequency time series (ffDTW, from dtw applied on ffts output), spectral parameters (SP, from specan). 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 (<marceloa27@gmail.com>). 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

```r
## 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")

# 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")

# alternatively we can provide our own SP matrix
Y <- specan(lbh_selec_table)

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

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

# add sound files and selec columns
Y <- data.frame(lbh_selec_table[, c(1, 3)], Y[, 1:2])

compare.methods(X = lbh_selec_table, methods = c("dfDTW"), custom1 = Y)

## End(Not run)

---

**consolidate**

**Consolidate (sound) files into a single directory**

**Description**

consolidate copies (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 for the files to be consolidated. Default is ".wav$" ignoring case.
- `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 NAs.

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.
Description

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

Usage

```r
coor.graph(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 (<marceloa27@gmail.com>)

References


Examples

```r
{ # load simulate singing events (see data documentation) data(sim.coor.sing) # ' # make coor.graphs in graphic device format}
```
Description

Monte Carlo randomization test to assess the statistical significance of overlapping or alternating singing (or any other simultaneously occurring behavior).

Usage

coor.test(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.

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

Logical. If TRUE removes the events that don’t have 2 interacting individuals. Default is FALSE.

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.

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 values. 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:

\[(\text{obs.overlap} - \text{mean.random.ovlp})/\text{mean.random.ovlp}\]

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 (<marceloa27@gmail.com>)
References


Examples

{
  # load simulated singing data (see data documentation)
  data(sim_coor_sing)

  # set global options
  # this can also be set within the function call
  warbleR_options(iterations = 100, pb = FALSE)

  # testing if coordination happens less than expected by chance
  coor.test(sim_coor_sing)

  # testing if coordination happens more than expected by chance
  coor.test(sim_coor_sing, less.than.chance = FALSE)

  # using "duration" method and "keep.song.order" as randomization procedure
  coor.test(sim_coor_sing, ovlp.method = "duration",
            randomization = "keep.song.order")
}


cut_sels

Cut selections into individual sound files

Description

cut_sels cuts selections from a selection table into individual sound files.

Usage

cut_sels(X, mar = 0.05, parallel = 1, path = NULL, dest.path = NULL, pb = TRUE,
         labels = c("sound.files", "selec"), overwrite = FALSE, ...)

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 signals (start and end). The output of manualoc or autodetect can be used as the input data frame.

**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.

**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.

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

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

**labels** String vector. Provides the column names that will be used as labels to create sound file names. Note that they should provide unique names (otherwise sound files will be overwritten). Default is c("sound.files","selec").

**overwrite** Logical. If TRUE sound files with the same name will be overwritten. Default is FALSE.

**...** Additional arguments to be passed to the internal writeWave function for customizing sound file output (e.g. normalization).

Details

This function allow users to produce individual sound files from the selections listed in a selection table as in lbh_selec_table.

Value

Sound files of the signals listed in the input data frame.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>) and Grace Smith Vidaurre

References


See Also

seltailor for tailoring selections blog post on cutting sound files
Examples

{  
  # save wav 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"))

  # cut selections
  cut_sels(lbh_selec_table, path = tempdir())

  #check this folder!!
  tempdir()
}

---

### dfDTW

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

#### Description

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

#### Usage

```r
dfDTW(X = NULL, wl = 512, wl.freq = 512, length.out = 20, wn = "hanning", ovlp = 70,
       bp = c(0, 22), threshold = 15, threshold.time = NULL, threshold.freq = NULL, img = TRUE,
       parallel = 1, path = NULL, ts.df = NULL, img.suffix = "dfDTW", pb = TRUE,
       clip.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 `manualoc` or `autodetec` can 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.
- **length.out**: A numeric vector of length 1 giving the number of measurements of dominant 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 c(0, 22).

**threshold**  Amplitude threshold (%) for dominant frequency detection. Default is 15.

**threshold.time**  Amplitude threshold (%) for the time domain. Use for 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.

**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`.

**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` 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`.

**frange.detec**  Logical. Controls whether frequency range of signal is automatically detected using the `frange.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.

**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`.
dfDTW

... Additional arguments to be passed to \texttt{trackfreqs} 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 \texttt{approx} function to interpolate values between dominant frequency measures. If 'img' is \texttt{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 \texttt{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 (<marceloa27@gmail.com>)

References


See Also

\texttt{speccreator} for creating spectrograms from selections, \texttt{snrspecs} for creating spectrograms to optimize noise margins used in \texttt{sig2noise} and \texttt{dfts, ffts, ffDTW} for frequency contour overlaid spectrograms. blog post on DTW similarity

Other spectrogram creators: \texttt{color.spectro, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, speccreator, trackfreqs}

Examples

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

  # run function  
dfDTW(lbh_selec_table, length.out = 30, flim = c(1, 12), bp = c(2, 9), wl = 300, path = tempdir())  
}
**dfts**

*Extract the dominant frequency values as a time series*

**Description**

dfts extracts the dominant frequency values as a time series of signals selected by manualoc or autodetec.

**Usage**

dfts(X, wl = 512, wl.freq = 512, length.out = 20, wn = "hanning", ovlp = 70, bp = c(0, 22), threshold = 0, threshold.time = NULL, threshold.freq = NULL, img = TRUE, parallel = 1, path = NULL, img.suffix = "dfts", pb = TRUE, clip.edges = FALSE, leglab = "dfts", frange.detec = FALSE, fsmooth = 0.1, raw.contour = FALSE, track.harm = FALSE, 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 manualoc or autodetec can 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.

- **length.out** A numeric vector of length 1 giving the number of measurements of dominant 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 c(0, 22). If 'frange' then the 'bottom.freq' and 'top.freq' columns are used bandpass limits.

- **threshold** amplitude threshold (%) for dominant frequency detection. Default is 0. Note that amplitude threshold for time and frequency domains can be defined independently. See "threshold.time" and "threshold.freq" arguments.

- **threshold.time** amplitude threshold (%) for the time domain. Use for 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.
**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.

**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.

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

**frange.detec** Logical. Controls whether frequency range of signal is automatically detected using the `frange.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.

**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).

**track.harm** Logical. If true warbleR’s `track_harm` function is used to track frequency contours. Otherwise seeawave’s `dfreq` is used by default.

**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 `trackfreqs`.

**Details**

This function extracts the dominant frequency values as a time series. The function uses the `approx` function to interpolate values between dominant frequency measures. If there are no frequencies above the amplitude threshold 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 time windows in between the signal in which amplitude was detected then the values of the adjacent will be interpolated to fill out the missing values (e.g. no NAs in between detected amplitude segments).

**Value**

The function returns a data frame with the dominant frequency values measured across the signals. If `raw.contour = TRUE` a list with the raw frequency detections (i.e. without interpolating values to make all contours of equal length) is returned. 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 dominant frequencies (see `trackfreqs` description for more details).
Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

sig2noise, trackfreqs, sp.en.ts.ffts, ffDTW, dfDTW

Other spectrogram creators: color.spectro, dfDTW, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts.speccreator, trackfreqs

Examples

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

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

ffDTW

Acoustic dissimilarity using dynamic time warping on fundamental frequency contours

Description

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

```
ffDTW(X, wl = 512, length.out = 20, wn = "hanning", ovlp = 70,
bp = c(0, 22), threshold = 5, img = TRUE, parallel = 1, path = NULL,
img.suffix = "ffDTW", pb = TRUE, clip.edges = TRUE, window.type = "none",
open.end = FALSE, scale = 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 of `manualoc` or `autodetect` can be used as the input data frame.
- **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 % 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 c(0, 22).
- **threshold**: Amplitude threshold (%) for fundamental frequency detection. Default is 5.
- **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. Default is NULL.
- **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 signal contour 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 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.
- **...**: Additional arguments to be passed to `trackfreqs` for customizing graphical output.
Details

This function extracts the fundamental frequency values as a time series and then calculates the pairwise acoustic dissimilarity of the selections using dynamic time warping. The function uses the approx function to interpolate values between fundamental 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 fundamental frequencies. Note that if no amplitude is detected at the beginning or end of the signals then NAs will be generated. On the other hand, if amplitude is not detected 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 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 fundamental frequencies.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

specreator for creating spectrograms from selections, snrspecs for creating spectrograms to optimize noise margins used in sig2noise
dfDTW dfts, ffts, dfDTW

Other spectrogram creators: color.spectro, dfDTW, dfts, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, specreator, trackfreqs

Examples

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

# run function
ffDTW(lbh_selec_table[1:4,], length.out = 30, flim = c(1, 12), img = TRUE, bp = c(1, 9), wl = 300, path = tempdir())
```
ffts extracts the fundamental frequency values as a time series of signals selected by manualoc or autodetec.

Usage

```r
ffts(X, wl = 512, length.out = 20, wn = "hanning", ovlp = 70, bp = c(0, 22),
threshold = 15, img = TRUE, parallel = 1, path = NULL, img.suffix = "ffts", pb = TRUE,
clip.edges = FALSE, leglab = "ffts", ff.method = "seewave", ...)
```

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 of manualoc or autodetec can be used as the input data frame.
- **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 % 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 c(0, 22).
- **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

ff.method

Details

This function extracts the fundamental frequency values as a time series. The function uses the `approx` function to interpolate values between fundamental frequency measures. If there are no frequencies above the amplitude threshold 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 `trackfreqs` description for more details).

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

See Also

`sig2noise, trackfreqs, dfts, ffDTW, dfDTW`

Other spectrogram creators: `color.spectro, dFTW, dfts, ffDTW, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, speccreator, trackfreqs`

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")) # save sound files

# run function
ffts(lbh_selec_table, length.out = 50, flim = c(1, 12), bp = c(2, 9), wl = 300, path = tempdir())
}
filtersels

Subset selection data frames based on manually filtered image files

Description

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

Usage

filtersels(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. The output of manualoc or autodetect can be used as the input
data frame.

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 pro-
duced by the function lspec. 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
speccreator). 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 (".wav") are included ( TRUE, de-
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. speccreator) 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 .wav files are ok, returns message "All files are ok!". Otherwise returns "These file(s) cannot be read" message with names of the corrupted .wav files.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

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"))

specreator(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 <- filtersels(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
lspec(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 <- filtersels(X = lbh_selec_table, lspec = TRUE,
path = tempdir())

## End(Not run)
```
Fix .wav files to allow importing them into R

Description

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

Usage

fixwavs(checksels = NULL, files = NULL, samp.rate = NULL, bit.depth = NULL, path = NULL, mono = FALSE, sox = FALSE)

Arguments

- checksels: Data frame with results from checksels. 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).
- sox: Logical indicating if SOX should be used for resampling. If TRUE SOX must be installed. Default is FALSE.

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. SOX must be installed to be able to run this function. 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 (<marceloa27@gmail.com>)

References


Examples

```r
## Not run:
# Save to temporary working directory

#check this folder
getwd()

## End(Not run)
```

---

**fix_extended_selection_table**

*Fix extended selection tables*

Description

`fix_extended_selection_table` fixes extended selection tables that have lost their attributes.

Usage

```r
fix_extended_selection_table(X, Y)
```

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`.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)
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"))

  # 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)
}
```

frange

 Detect frequency range iteratively

Description

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

Usage

```r
frange(X, wl = 512, it = "jpeg", line = TRUE, fsmooth = 0.1, threshold = 10, 
dB.threshold = NULL, wn = "hanning", flim = c(0, 22), 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 .wav files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of manualoc or autodetect can also be used as the input data frame.
frange

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 help 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 c(0, 22).

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.

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.

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 \texttt{spectro}. Default is 50. This is used for calculating the frequency spectrum (using \texttt{meanspec}) and producing the spectrogram (using \texttt{spectro}, if \texttt{img = TRUE}).

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} = \texttt{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 \texttt{NULL}.

img  Logical. Controls whether a plot is produced. Default is \texttt{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 \texttt{NULL} (default) then the current working directory is used.

pb  Logical argument to control progress bar and messages. Default is \texttt{TRUE}.

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

**Details**

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

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

If \texttt{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 \texttt{frange.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 \texttt{img = TRUE} (see details).

**Author(s)**

Marcelo Araya-Salas (<marceloa27@gmail.com>)
References


See Also

frange.detec, autodetec

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"))  

frange(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())  
}
```

frange.detec  Detect frequency range on wave objects

Description

frange.detec detects the frequency range of acoustic signals on wave objects.

Usage

```
frange.detec(wave, wl = 512, fsmooth = 0.1, threshold = 10,  
             dB.threshold = NULL, wn = "hanning", flim = c(0, 22), 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 helps to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.
frange.detec

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 c(0, 22).

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).

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

- 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 (<marceloa27@gmail.com>)

References


See Also

frange, autodetec

Examples

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

data(sheep) frange.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")
}

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

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 (<marceloa27@gmail.com>)

References


See Also
dfts, trackfreqs.

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"))
  writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))
  
  # measure frequency contours
  dom.freq.ts <- dfts(X = lbh_selec_table, path = tempdir())
  
  # get number of inflections
  inflections(X = dom.freq.ts)
}
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 checksels
- 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 (<marceloa27@gmail.com>)

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**

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

# Check if lbh_selec_table is an extended selection table
is_extended_selection_table(lbh_selec_table)

# Write four different 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"))

# Create a selection table with extended options
st <- selection_table(lbh_selec_table, extended = TRUE, confirm.extended = FALSE, path = tempdir())

# Check if st is an extended selection table
is_extended_selection_table(st)

# Print the class of 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**

```r
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 `checksels`

**Value**

A logical argument indicating whether the object class is `selection_table`

**Author(s)**

Marcelo Araya-Salas (<marceloa27@gmail.com>)
lbh_selec_table

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"

Examples

{
  # 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 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

data(lbh_selec_table)

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  
**sel.comment**  selection comments  
**rec.comment**  recording comments

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 'lbh_selec_table'. 'lbh_selec_table' will be removed in future package version.

Source

Marcelo Araya Salas, warbleR

---

### lspec

*Create long spectrograms of whole sound files*

**Description**

lspec produces image files with spectrograms of whole sound files split into multiple rows.

**Usage**

```r
lspec(X = NULL, flim = c(0, 22), 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)
```

**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, two red dotted lines are plotted at the start and end of a selection and the selections are labeled with the selection number (and selection comment, if available). Default is NULL.

- **flim**  
  A numeric vector of length 2 indicating the highest and lowest frequency limits (kHz) of the spectrogram, as in spectro. Default is c(0, 22).

- **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).
ovep 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.

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 "tif" and "jpeg" are admitted. Default is "jpeg".

flike 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.

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.

directional 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_param). If supplied then lines above the selections belonging to the same ’song’ are plotted. Ignored if ’X’ is not provided.

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 (<marceloa27@gmail.com>)

References


See Also

lspec2pdf, catalog2pdf, blog post on spectrogram pdfs

Examples

```r
# Not run:
# Save 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"))

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

# including selections
lspec(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)
```

---

**lspec2pdf**

`lspec2pdf` combines `lspec` images in .jpeg format to a single pdf file.

**Description**

`lspec2pdf` combines `lspec` images in .jpeg format to a single pdf file.

**Usage**

```
lspec2pdf(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 `lspec` function into a single pdf (for each sound file).

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

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

# now create single pdf removing jpeg
lspec2pdf(keep.img = FALSE, path = tempdir())

# check this folder
tempdir()

## End(Not run)
```
manualoc

Interactive view of spectrograms

Description

manualoc produces an interactive spectrographic view in which the start and end times of acoustic signals can be measured.

Usage

```r
manualoc(wl = 512, flim = c(0,12), seltime = 1, tdisp = NULL, reccomm =
FALSE, wn = "hanning", title = TRUE, selcomm = FALSE, osci = FALSE, player =
NULL, pal = reverse.gray.colors.2, path = NULL, flist = NULL,
fast.spec = FALSE, ext.window = TRUE, width = 15, height = 5)
```

Arguments

- **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(0,12).
- **seltime**: A numeric vector of length 1 indicating the time interval in seconds at which the spectrograms are produced with higher resolution (ovlp = 70) and oscillograms (if osci = TRUE). Default is 1 second.
- **tdisp**: A numeric vector of length 1 specifying the length in seconds of the total sound file to be displayed. Default is NULL which displays the full sound file.
- **reccomm**: Logical argument. If TRUE pops up a comment window at the end of each sound file. The comment needs to be quoted. Default is FALSE.
- **wn**: A character vector of length 1 specifying the window function (by default "hanning"). See function `ftwindow` for more options.
- **title**: Logical argument. If TRUE the name of the sound file will be printed as the main title of the spectrogram window. Default is TRUE.
- **selcomm**: Logical argument. If TRUE pops up a comment window after each selection. The comment is printed as a label on the selected unit. The comment must be quoted. Default is FALSE.
- **osci**: Logical argument. If TRUE adds a oscillogram whenever the spectrograms are produced with higher resolution (see seltime). Default is FALSE.
- **player**: Path to or name of a program capable of playing a wave file by invocation from the command line. If under Windows and no player is given, windows player will be chosen as the default. "vlc" works in Linux if vlc player is installed. 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.
Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.

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

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.

Logical. If TRUE then and external graphic window is used. Default dimensions can be set using the 'width' and 'height' arguments. Default is TRUE.

Numeric of length 1 controlling the width of the external graphic window. Ignored if ext.window = FALSE. Default is 15.

Numeric of length 1 controlling the height of the external graphic window. Ignored if ext.window = FALSE. Default is 5.

This function may work very slowly with middle and large size sound files. We strongly suggest using other software tools (e.g. Raven, Avisoft) to create selection tables manually.

Users can zoom-in a specific sound file segment by clicking at the start and end (left side and right side) of the segment. To select the start and end of a vocalization unit the users need to click at the end and then at the start (right side and left side) of the unit. In addition, 6 "buttons" are provided at the upper right side of the spectrogram that allow to display a full view of the spectrogram ("Full view"), go back to the previous view ("Previous view"), stop the analysis ("Stop"), go to the next sound file ("Next rec"), play the current view using external software ("Play", see "player" argument), or delete the last manual selection in the current sound file ("Delete"). When a unit has been selected, the function plots a red circle with the selection number in the middle point of the selection in the spectrogram. It also plots vertical dotted lines at the start and end of the selection. The circle and lines "disappear" when the selection is deleted ("Delete" button). Only the last selection can be deleted.

The function produces a .csv file (manualoc_output.csv) with information about the .wav file name, selection number, start and end time, selection comment (selcomm), and sound file comment (reccomm). The file is saved in the working directory and is updated every time the user moves into the next sound file (Next rec "button") or stop the process (Stop "button"). When resuming the process (after "stop" and re-running the function in the same working directory), the function will keep the previous selections and will only pick up .wav files that are not present in the .csv file (not previously analyzed). When users go to the next sound file (Next rec "button") without making any selection the file is still included in the .csv file, with NA's in the "end", "time" and "select" field.

Windows length (wl) controls the temporal and frequency precision of the spectrogram. A high "wl" value increases the frequency resolution but reduces the temporal resolution, and vice versa. Any color palette that comes with the seewave package can be used: temp.colors, reverse.gray.colors.1, reverse.gray.colors.2, reverse.heat.colors, reverse.terrain.colors, reverse.topo.colors, reverse.cm.colors, heat.colors, terrain.colors, topo.colors, cm.colors. The function is slow when working on files of
length > 5min. In most cases other sound analysis software for manually selecting acoustic signals (e.g. Raven, Syrinx) should be preferred.

Value

csv file saved in the working directory with start and end time of selections.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

See Also

seltailor

Examples

## Not run:
# save wav file examples
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4"))
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav"))
writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav"))
writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))

manualoc(wl = 300, path = tempdir())
# need to use the buttons to manipulate function
# check working directory for .csv file after stopping function
# check here:
tempdir()

## End(Not run)

### mfcc_stats

**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 .wav files, 2) "sel": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of manualoc or autodetect can be used as the input data frame.

\(\text{ovlp}\) 
Numeric vector of length 1 specifying \% of overlap between two consecutive windows. Internally this is used to set the 'hoptime' argument in \texttt{melfcc}. Default is 50.

\(\text{wl}\) 
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.

\(\text{bp}\) 
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.

\(\text{path}\) 
Character string containing the directory path where the sound files are located.

\(\text{numcep}\) 
Numeric vector of length 1 controlling the number of cepstra to return (see \texttt{melfcc}).

\(\text{nbands}\) 
Numeric vector of length 1 controlling the number of warped spectral bands to use (see \texttt{melfcc}). Default is 40.

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

\(\text{pb}\) 
Logical argument to control progress bar and messages. Default is \texttt{TRUE}.

\(...\) 
Additional parameters to be passed to \texttt{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 (<marceloa27@gmail.com>
References


See Also

fixwavs, rm_sil,

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_st)
}
```
move.imgs

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 (<marceloa27@gmail.com>)

References


See Also

- filtersels

Other data manipulation: open_wd, split_wavs

Examples

```{r}
data("Crypt.soui")
writeWave(Crypt.soui, file.path(tempdir(), "Crypt.soui.wav")) #save sound files

ad <- autodetect(threshold = 6, bp = c(1, 3), mindur = 1.2,
maxdur = 3, img = FALSE, ssmooth = 600, wl = 300, flist = "Crypt.soui.wav", path = tempdir())
```
#track dominant frequency graphs with freq range detection
trackfreqs(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90, it = "tiff",
bp = c(1, 2), 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(),
to = file.path(tempdir(), "imgs"), overwrite = TRUE)

# Check this folder
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. Downsampling is done using the
resample function from the bioacoustics package (which should be installed),
which seems to generate aliasing. This can be avoided by downsampling af-
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 num-
ber 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.
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.

Value

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

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>) and Grace Smith Vidaurre

References


Examples

```r
## Not run:
# download mp3 files from xeno-canto
querxc(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

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 trackfreqs 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 (<marceloa27@gmail.com>)
References


See Also

speccreator for creating spectrograms from selections, snrspecs for creating spectrograms to optimize noise margins used in sig2noise and dfts, ffts, ffDTW for frequency contour overlaid spectrograms. blog post on DTW similarity

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, phylo_spectro, snrspecs, sp.en.ts, speccreator, trackfreqs

Examples

```r
## Not run:
#load data
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "lbh_selec_table"))

writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) #save sound files
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
df <- df_ts(X = lbh_selec_table, threshold = 10, img = FALSE, path = tempdir())
se <- se_ts(X = lbh_selec_table, threshold = 10, img = FALSE, path = tempdir())

# run function
multi_DTW(df, se)

## End(Not run)
```

---

new_function_names

Data frame detailing function name changes

Description

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

Usage

data(new_function_names)

Format

An object of class data.frame with 26 rows and 2 columns.
open_wd

Author(s)
Marcelo Araya-Salas (<marceloa27@gmail.com>)

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.

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 (<marceloa27@gmail.com>)

References
Araya-Salas, M., & Smith-Vidaurre, G. (2017). warbleR: An R package to streamline analysis of

See Also
move.imgs
Other data manipulation: move.imgs, split_wavs

Examples
{
  open_wd()
}
ovlp_sels  Find overlapping selections

Description

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

Usage

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

Arguments

X
'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. The output of manualoc or autodetec can be used as the input data frame. Other data frames can be used as input, but must have at least the 4 columns mentioned above.

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 selections 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 Logical to control if labels are reused across different sound files (if TRUE, default).

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).
phylo_spectro

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.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

`filtersels lbh_selec_table`

Examples

```{r}
# no overlap
ovlp_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
ovlp_sels(X = Y)

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

# get index instead
ovlp_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`.  

---

**Description**

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

---

**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.

**Author(s)**

Marcelo Araya-Salas (<marceloa27@gmail.com>)

**References**


**See Also**

`filtersels lbh_selec_table`

**Examples**

```{r}
# no overlap
ovlp_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
ovlp_sels(X = Y)

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

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

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).
- **tree**: 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).
- **type**: Character string of length 1 specifying the type of phylogeny to be drawn (as in plot.phylo). Only 'phylogram' (default) and 'fan' are allowed.
- **par.mar**: 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.
- **size**: 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.
- **offset**: Numeric vector of length 1 controlling the space between tips and spectrograms. Default is 0.
- **path**: Character string containing the directory path where the sound files are located. If NULL (default) then the current working directory is used.
- **ladder**: 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.
- **horizontal**: Logical. Controls whether spectrograms in a fan phylogeny are place 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 (speccreator) or phylogeny plotting function (plot.phylo) for customizing graphical output. Only rightwards phylogenies can be plotted.

Details

The function add 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 speccreator 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 (<marceloa27@gmail.com>)


specreator, plot.phylo

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, multi_DTW, snrspecs, sp.en.ts, specreator, trackfreqs

# First set empty folder

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 <- xcorr(X, bp = c(1, 10))
xc.tree <- 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)

---

**querxc**

**Access 'Xeno-Canto' recordings and metadata**

**Description**

querxc downloads recordings and metadata from Xeno-Canto.

**Usage**

```r
querxc(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 downloaded (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 xcmaps.

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 (<marceloa27@gmail.com>)

References

**read_wave**

A wrapper for tuneR’s `readWave` function that read sound files listed within selection tables.

### Description

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

### Usage

```r
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). '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). If NULL or higher than the number of channels in a wave then the first channel is used.

- **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.

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 (<marceloa27@gmail.com>)

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"))

warbleR::read_wave(X = lbh_selec_table, index = 1, path = tempdir())
```
Description

`resample_est` changes sampling rate and bit depth of wave objects in an extended selection table.

Usage

```r
resample_est(X, samp.rate = 44.1, bit.depth = 16,
sox = FALSE, avoid.clip = TRUE, pb = FALSE, parallel = 1)
```

Arguments

- **X**: object of class 'extended_selection_table' (see `selection_table`).
- **samp.rate**: Numeric vector of length 1 with the sampling rate (in kHz) for output files. Default is NULL.
- **bit.depth**: Numeric vector of length 1 with the dynamic interval (i.e., bit depth) for output files.
- **sox**: Logical to control whether SOX is used internally for resampling. Sox must be installed. Default is FALSE. SOX is a better option if having aliasing issues after resampling.
- **avoid.clip**: Logical to control whether the volume is automatically adjusted to avoid clipping high amplitude samples when resampling. Ignored if `sox = FALSE`. Default is TRUE.
- **pb**: Logical argument to control progress bar. 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).

Details

This function aims to simplify the process of homogenizing sound files (sampling rate and bit depth). This is a necessary step before running any further (bio)acoustic analysis. Either SOX (if `sox = TRUE`) or the `bioacoustics` package (if `sox = FALSE`) should be installed.

Value

An extended selection table with the modified wave objects.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>) #last modification on oct-15-2018 (MAS)

References

### rm_channels

**rm_channels** remove channels in wave files

**Description**

rm_channels remove channels in wave files

**Usage**

```r
rm_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.

---

### Examples

```r
## Not run:
data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4", "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
X <- selection_table(X = lbh_selec_table, extended = TRUE, confirm.extended = FALSE, pb = FALSE, path = tempdir())

# resample
Y <- resample_est(X)

## End(Not run)
```
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 (<marceloa27@gmail.com>)

References


See Also

fixwavs, rm_sil,

Examples

{  
  # 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"))

  rm_channels(channels = 1, path = tempdir())

  #check this folder
  tempdir()
  }

rm_sil

Remove silence in wave files

Description

rm_sil Removes silences in wave files
Usage

```
rm_sil(path = NULL, min.sil.dur = 2, img = TRUE, it = "jpeg", flim = c(0, 12),
      flist = NULL, parallel = 1, pb = TRUE)
```

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 c(0,12). Ignored if 'img = FALSE'.
- **flist** 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.
- **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 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". If 'img = TRUE' then spectrogram images highlighting the silence segments that were removed are also saved.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

fixwavs, autodetect
Examples

```r
# save sound file examples
data(list = c("Phae.long1", "Phae.long2","lbh_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
ev(wv1)

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

#remove silence
rm_sil(flist = "wv1.wav", pb = FALSE, path = tempdir())

#check this floder
tempdir()
```

selec.table

**Alternative name for** 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**

Simulated coordinated singing events.

**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, ...)

Arguments

X data frame with the following columns: 1) "sound.files": name of the .wav files, 2) "selec": 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 to double check selections. The output of manualoc or autodetec can be used as the input object for other warbleR functions.

max.dur the maximum duration of expected for a selection (ie. 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 = wavdur()'. 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).

...  Additional arguments to be passed to checksels for customizing checking routine.

Details

This function creates and 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. specan, xcorr, catalog, dfDTW). Note also that corrupt files can be fixed using fixwavs (’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 checksels 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 (<marceloa27@gmail.com>)

**References**


**See Also**

checkwavs, blog post on extended selection tables

**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"))  

# make selection table  
st <- selection_table(X = lbh_selec_table, path = tempdir())  
is_selection_table(st)  

#' # make extended selection table  
st <- selection_table(X = lbh_selec_table, extended = TRUE,  
                      confirm.extended = FALSE,  
                      path = tempdir())  
is_extended_selection_table(st)  

### make extended selection by song  
# create a song variable  
lbh_selec_table$song <- as.numeric(lbh_selec_table$sound.files)  
st <- selection_table(X = lbh_selec_table, extended = TRUE,  
                      confirm.extended = FALSE, by.song = "song", path = tempdir())  
}
```
seltailor produces an interactive spectrographic view (similar to manualoc) in which the start/end times and frequency range of acoustic signals listed in a data frame can be adjusted.

Usage

```
seltailor(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' 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. The output of manualoc or autodetect can be used as the input data frame. Other data frames can be used as input, but must have at least the 4 columns mentioned above. 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 and 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"), go to the next sound file ("next"), return to the previous selection ("previous") or delete the current selection ("delete"). An additional "button" ("contour") to tailored frequency contour is shown when ‘ts.df’ is provided. 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 (next sel "button") 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 'next' button) 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. 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 (<marceloa27@gmail.com>)

References

See Also
manualoc

Examples
## 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"))
seltailor(X = lbh_selec_table, flim = c(1,12), wl = 300, auto.next = TRUE, path = tempdir())
# Read output .csv file
```r
seltailor.df <- read.csv(file.path(tempdir(), "seltailor_output.csv"))

# check this directory for .csv file after stopping function
tempdir()

## End(Not run)
```

### sig2noise

**Measure signal-to-noise ratio**

**Description**

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

**Usage**

```r
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). The output of `manual_loc` can also be used as the input data frame.
- `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 (mean(env(S))/mean(env(N)))
  - 2: ratio of S amplitude envelope quadratic mean to N amplitude envelope quadratic mean (rms(env(S))/rms(env(N)))
  - 3: ratio of the difference between S amplitude envelope quadratic mean and N amplitude envelope quadratic mean to N amplitude envelope quadratic mean ((rms(env(S)) -rms(env(N)))/rms(env(N)))
```
sig2noise

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. snrspecs can be used to troubleshoot different noise margins.

Value

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

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>) and Grace Smith Vidaurre

References


Examples

```{ data(list = c("Phae.long1","1bh_selec_table")) ```
writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav")) # save sound files

# specifying the correct margin is important
# use snrspecs 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())

Description

sim.coor.sing are 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). Will be replaced by `sim_coor_sing` in future package versions.

Usage

data(sim.coor.sing)

Format

**sim.coor.sing**  Simulated coordinated singing events.

**sim_coor_sing**  Alternative name for **sim.coor.sing**. **sim.coor.song** will be deprecated in future versions.

Usage

**sim_coor_sing**

Format

An object of class data.frame with 446 rows and 4 columns.
**sim_songs**

*Simulate animal vocalizations*

**Description**

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

**Usage**

`sim_songs(n = 1, durs = 0.2, harms = 3, amps = c(1, 0.5, 0.2), 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.
- **amps**: Numeric vector with the relative amplitude of each of the harmonics (including the fundamental frequency).
- **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 of length 1 defining the sigma value of the brownian motion model. Higher values will produce faster frequency modulations. Ignored if `diff_fun == "BB"`. Default is 0.1. Check the `BB` 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.
sim_songs

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diff_fun</td>
<td>Character vector of length 1 controlling the function used to simulate the brownian motion process of frequency drift across time. Only &quot;BB&quot; and &quot;GBM&quot; are accepted at this time. Check the BB for more details.</td>
</tr>
<tr>
<td>fin</td>
<td>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.</td>
</tr>
<tr>
<td>fout</td>
<td>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.</td>
</tr>
<tr>
<td>shape</td>
<td>Character string of length 1 controlling the shape of in and out amplitude fading of the song sub-units ('fin' and 'fout'). &quot;linear&quot; (default), &quot;exp&quot; (exponential), and &quot;cos&quot; (cosine) are currently allowed.</td>
</tr>
<tr>
<td>selec_table</td>
<td>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 saved as a &quot;.wav&quot; file in the working directory. Default is FALSE.</td>
</tr>
<tr>
<td>file_name</td>
<td>Character string for naming the &quot;.wav&quot; file. Ignored if 'selec_table' is FALSE. If not provided the date-time stamp will be used.</td>
</tr>
<tr>
<td>path</td>
<td>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.</td>
</tr>
</tbody>
</table>

Details

This function uses a brownian motion stochastic process to simulate animal vocalizations (i.e. frequency traces across time). 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 (<marceloa27@gmail.com>)

References


See Also

querxc for for downloading bird vocalizations from an online repository.
Examples
{
  # simulate a song with 3 elements and no harmonics
  sm_sng <- sim_songs(n = 3, harms = 1)

  # plot spectro
  seewave::spectro(sm_sng)

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

  # plot spectro
  seewave::spectro(sm_sng2)
}

snrspecs

Spectrograms with background noise margins

Description

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

Usage

snrspecs(X, wl = 512, flim = c(0, 22), 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,
         snrmar = 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). The output of manual_loc 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.
flim   A numeric vector of length 2 for the frequency limit in kHz of the spectrogram, as in spectro. Default is c(0, 22).
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.
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inner.mar</td>
<td>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.</td>
</tr>
<tr>
<td>outer.mar</td>
<td>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.</td>
</tr>
<tr>
<td>picsize</td>
<td>Numeric argument of length 1, controls relative size of spectrogram. Default is 1.</td>
</tr>
<tr>
<td>res</td>
<td>Numeric argument of length 1 that controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.</td>
</tr>
<tr>
<td>cexlab</td>
<td>Numeric vector of length 1 specifying relative size of axis labels. See spectro.</td>
</tr>
<tr>
<td>title</td>
<td>Logical argument to add a title to individual spectrograms. Default is TRUE.</td>
</tr>
<tr>
<td>before</td>
<td>Logical. If TRUE noise is only measured right before the signal (instead of before and after). Default is FALSE.</td>
</tr>
<tr>
<td>eq.dur</td>
<td>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.</td>
</tr>
<tr>
<td>propwidth</td>
<td>Logical argument to scale the width of spectrogram proportionally to duration of the selected call. Default is FALSE.</td>
</tr>
<tr>
<td>xl</td>
<td>Numeric vector of length 1, a constant by which to scale spectrogram width if propwidth = TRUE. Default is 1.</td>
</tr>
<tr>
<td>osci</td>
<td>Logical argument to add an oscillogram underneath spectrogram, as in spectro. Default is FALSE.</td>
</tr>
<tr>
<td>gr</td>
<td>Logical argument to add grid to spectrogram. Default is FALSE.</td>
</tr>
<tr>
<td>sc</td>
<td>Logical argument to add amplitude scale to spectrogram, default is FALSE.</td>
</tr>
<tr>
<td>mar</td>
<td>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.</td>
</tr>
<tr>
<td>snrmar</td>
<td>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.</td>
</tr>
<tr>
<td>it</td>
<td>A character vector of length 1 giving the image type to be used. Currently only &quot;tif&quot; and &quot;jpeg&quot; are admitted. Default is &quot;jpeg&quot;.</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>
</tbody>
</table>

**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.
snrspecs

Value

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

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>) and Grace Smith Vidaurre

References


See Also

trackfreqs for creating spectrograms to visualize frequency measurements by specan, speccreator for creating spectrograms after using manualoc

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, sp.en.ts, speccreator, trackfreqs

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()

snrspecs(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", wl = 300,
path = tempdir())

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

snrspecs(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, wl = 300, path = tempdir())

#check this folder!
tempdir()

## End(Not run)
```
song_param

Calculates acoustic parameters at the song level

Description

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

Usage

song_param(X = NULL, song_colm = "song", mean_colm = NULL, min_colm = NULL, max_colm = NULL, elm_colm = NULL, elm_fun = NULL, sd = FALSE, parallel = 1, pb = TRUE, 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 .wav files, 2) "select": number of the selections, 3) "start": start time of selections, 4) "end": end time of selections. The output of manualoc or autodetec can be used as the input data frame. Other data frames can be used as input, but must have at least the 4 columns mentioned above.

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)
- **gap.duration**: 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 (<marceloa27@gmail.com>)

References


See Also

specan
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"))

# add a 'song' column
lbh_selec_table$song <- rep(1:4, each = 3)[1:11]

# measure acoustic parameters
sp <- specan(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"))

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

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

# caculate song-level parameters for selecting parameters with mean_colm, max_colm
# and min_colm and weighted by duration
song_param(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_param(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_param(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 (<marceloa27@gmail.com>)

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_colms(X = selec.table)

#check names again
names(selec.table)

sp.en.ts Extract the spectral entropy across signals as a time series

Description

sp.en.ts spectral entropy across signals as a time series. of signals selected by manualoc or sp.en.ts.

Usage

sp.en.ts(X, wl = 512, length.out = 20, wn = "hanning", ovlp = 70, bp = "frange", threshold = 15, img = TRUE, parallel = 1, path = NULL, img.suffix = "sp.en.ts", pb = TRUE, clip.edges = FALSE, leglab = "sp.en.ts", sp.en.range = c(2, 10), ...)
sp.en.ts

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 manualoc or autodetect 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. Note that this is particularly important for measuring spectral entropy. Low values (~100) generate a very detail contour of the variation in spectral entropy that is probably not useful for assessing signal similarity.

length.out: A character vector of length 1 giving the number of measurements of spectral entropy 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). If 'frange' (default) then the 'bottom.freq' and 'top.freq' columns are used bandpass limits.

threshold: amplitude threshold (%) for dominant 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.

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.

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

sp.en.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 trackfreqs for customizing graphical output.

Details

This function spectral entropy across signals as a time series. The function uses the approx function to interpolate values between spectral entropy measures (calculated with csh). If there are no frequencies above the amplitude threshold 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). Missing values at the start of end can be removed with "clip.edges".

Value

A data frame with the dominant 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 dominant frequencies (see trackfreqs description for more details).

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

speccreator for creating spectrograms from selections, snrspecs for creating spectrograms to optimize noise margins used in sig2noise

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, speccreator, trackfreqs

Examples

{
  #load data
  data(list = c("Phae.long1", "Phae.long2", "Phae.long3", "Phae.long4","lbh_selec_table"))
  writeWave(Phae.long2, file.path(tempdir(), "Phae.long2.wav")) #save sound files
  writeWave(Phae.long1, file.path(tempdir(), "Phae.long1.wav"))
  writeWave(Phae.long3, file.path(tempdir(), "Phae.long3.wav")) #save sound files
  writeWave(Phae.long4, file.path(tempdir(), "Phae.long4.wav"))

  # without clip edges
  sp.en.ts(X = lbh_selec_table, threshold = 10, clip.edges = FALSE, length.out = 10,
           type = "b", sp.en.range = c(-25, 10), path = tempdir())

  # with clip edges and length.out 10
  sp.en.ts(X = lbh_selec_table, threshold = 10, bp = c(2, 12), clip.edges = TRUE, length.out = 10,
           path = tempdir())
}

specan

Measure acoustic parameters in batches of sound files

Description

specan measures acoustic parameters on acoustic signals for which the start and end times are provided.

Usage

specan(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, nharm = 3, ...)

Arguments

X '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, 3) "start": start time of selections, 4) "end": end time of selections. The output of manualoc or autodetec 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.
Character vector of length 1 specifying window name. Default is hanning'. See function `ftwindow` for more options.

A numeric vector of length 1 to smooth the frequency spectrum with a mean sliding window (in kHz) used for mean peak frequency detection. This help to average amplitude "hills" to minimize the effect of amplitude modulation. Default is 0.1.

Logical. If `TRUE` harmonicity related parameters (fundamental frequency parameters [meanfun, minfun, maxfun], hn_freq, hn_width, harmonics and HNR) are measured. Note that measuring these parameters considerably increases computing time.

Numeric vector of length 1 setting the number of harmonics to analyze.

Additional parameters to be passed to `analyze`, which measures parameters related to harmonicity.

The output of `manualoc` or `autodetect` can be used directly without any additional modification. 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 seeawave 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.

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`
- **meanfun**: average of fundamental frequency measured across the acoustic signal
- **minfun**: minimum fundamental frequency measured across the acoustic signal
- **maxfun**: maximum fundamental frequency measured across the acoustic signal
- **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.
- **hn_freq**: mean frequency of the 'n' upper harmonics (kHz) (see `analyze`). Number of harmonics is defined with the argument 'nharmonics'.
- **hn_width**: mean bandwidth of the 'n' upper harmonics (kHz) (see `analyze`). Number of harmonics is defined with the argument 'nharmonics'.
- **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'.
- **HNR**: harmonics-to-noise ratio (dB). A measure of the harmonic content generated by `getPitchAutocor`. 
Description

`specreator` creates spectrograms of signals from selection tables.

Usage

```r
specreator(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 = adjustcolor("#E37222", 0.6), 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. The output of `manualoc` or `autodetect` can be used as the input data.
frame. 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.

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 red 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 'adjustcolor("red2", alpha.f = 0.7)'.

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, delimiting 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. Ignored if 'by.song' is NULL. 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 cut sound 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 spectro, so it takes the same arguments.

Details

This function provides access to batch process of (a modified version of) the spectro function from the '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 (<marceloa27@gmail.com>) and Grace Smith Vidaurre
spec_param

References


See Also

trackfreqs for creating spectrograms to visualize frequency measurements by specan, snrspecs for creating spectrograms to optimize noise margins used in sig2noise

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, trackfreqs

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
  speccreator(X = lbh_selec_table, flim = c(0, 11), res = 300, mar = 0.05, wl = 300, path = tempdir())

  # check this folder
  tempdir()
}

spec_param

Plot a mosaic of spectrograms with varying display parameters

Description

spec_param plots a mosaic of spectrograms with varying display parameters to facilitate selection of display parameters

Usage

spec_param(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
length.out

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.

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 spec_param.

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 warsbleR and seewave 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 (<marceloa27@gmail.com>)
spec_param

See Also

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

Examples

## Not run:
# Save to temporary working directory

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

# variable collevels
spec_param(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
spec_param(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
spec_param(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
spec_param(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
spec_param(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
spec_param(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 folder
split_wavs

Description

split_wavs splits sound files in shorter segments

Usage

split_wavs(path = NULL, sgmt.dur = 10, sgmts = NULL, files = NULL, parallel = 1, pb = TRUE)

Arguments

- **path**: Directory path where sound files are found. If NULL (default) then the current working directory is used.
- **sgmt.dur**: Numeric. Duration (in s) of segments in which sound files would be split. Sound files shorter than `sgmt.dur` won’t be split. Ignored if `sgmts` is supplied.
- **sgmts**: Numeric. Number of segments in which to split each sound file. If supplied `sgmt.dur` is ignored.
- **files**: Character vector indicating the subset of files that will be split.
- **parallel**: Numeric. Controls whether parallel computing is applied. It specifies the number of cores to be used. Default is 1 (i.e. no parallel computing).
- **pb**: Logical argument to control progress bar. Default is TRUE.

Details

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

Value

Wave files for each segment in the working directory (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 cuts (sound.files) and the start and end of cuts in the original files.

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References

trackfreqs

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_wavs(sgmt.dur = 1, path = tempdir())

  # Check this folder
  tempdir()
}

trackfreqs  
Spectrograms with frequency measurements

Description

trackfreqs creates spectrograms to visualize dominant and fundamental frequency measurements (contours) of signals selected by manualoc or autodetec.

Usage

trackfreqs(X, wl = 512, wl.freq = 512, flim = c(0, 22), 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 = c(0, 22), cex = c(0.6, 1), threshold = 15, threshold.time = NULL, 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 (selec), and start and end time of signal (start and end). The output of manualoc or autodetec can also be used as the input data frame.
### trackfreqs

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wl</code></td>
<td>A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.</td>
</tr>
<tr>
<td><code>wl.freq</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>flim</code></td>
<td>A numeric vector of length 2 for the frequency limit of the spectrogram (in kHz), as in <code>spectro</code>. Default is <code>c(0, 22)</code>.</td>
</tr>
<tr>
<td><code>wn</code></td>
<td>Character vector of length 1 specifying window name. Default is &quot;hanning&quot;. See function <code>ftwindow</code> for more options.</td>
</tr>
<tr>
<td><code>pal</code></td>
<td>A color palette function to be used to assign colors in the plot, as in <code>spectro</code>. Default is <code>reverse.gray.colors.2</code>.</td>
</tr>
<tr>
<td><code>ovlp</code></td>
<td>Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in <code>spectro</code>. Default is 70.</td>
</tr>
<tr>
<td><code>inner.mar</code></td>
<td>Numeric vector with 4 elements, default is <code>c(5,4,4,2)</code>. Specifies number of lines in inner plot margins where axis labels fall, with form <code>c(bottom, left, top, right)</code>. See <code>par</code>.</td>
</tr>
<tr>
<td><code>outer.mar</code></td>
<td>Numeric vector with 4 elements, default is <code>c(0,0,0,0)</code>. Specifies number of lines in outer plot margins beyond axis labels, with form <code>c(bottom, left, top, right)</code>. See <code>par</code>.</td>
</tr>
<tr>
<td><code>picsize</code></td>
<td>Numeric argument of length 1. Controls relative size of spectrogram. Default is 1.</td>
</tr>
<tr>
<td><code>res</code></td>
<td>Numeric argument of length 1. Controls image resolution. Default is 100 (faster) although 300 - 400 is recommended for publication/presentation quality.</td>
</tr>
<tr>
<td><code>cexlab</code></td>
<td>Numeric vector of length 1 specifying the relative size of axis labels. See <code>spectro</code>.</td>
</tr>
<tr>
<td><code>title</code></td>
<td>Logical argument to add a title to individual spectrograms. Default is <code>TRUE</code>.</td>
</tr>
<tr>
<td><code>propwidth</code></td>
<td>Logical argument to scale the width of spectrogram proportionally to duration of the selected call. Default is <code>FALSE</code>.</td>
</tr>
<tr>
<td><code>xl</code></td>
<td>Numeric vector of length 1. A constant by which to scale spectrogram width. Default is 1.</td>
</tr>
<tr>
<td><code>osci</code></td>
<td>Logical argument to add an oscillogram underneath spectrogram, as in <code>spectro</code>. Default is <code>FALSE</code>.</td>
</tr>
<tr>
<td><code>gr</code></td>
<td>Logical argument to add grid to spectrogram. Default is <code>FALSE</code>.</td>
</tr>
<tr>
<td><code>sc</code></td>
<td>Logical argument to add amplitude scale to spectrogram, default is <code>FALSE</code>.</td>
</tr>
<tr>
<td><code>bp</code></td>
<td>A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or &quot;frange&quot; to indicate that values in <code>bottom.freq</code> and <code>top.freq</code> columns will be used as bandpass limits. Default is <code>c(0, 22)</code>.</td>
</tr>
<tr>
<td><code>cex</code></td>
<td>Numeric vector of length 2, specifies relative size of points plotted for frequency measurements and legend font/points, respectively. See <code>spectro</code>.</td>
</tr>
<tr>
<td><code>threshold</code></td>
<td>Amplitude threshold (%) for fundamental and dominant frequency detection as well as frequency range from the spectrum (see &quot;frange.detec&quot;). Default is 15. WILL BE DEPRECATED. Use 'threshold.time' and 'threshold.time' instead.</td>
</tr>
</tbody>
</table>
trackfreqs

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.

custom.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...
Trackfreqs

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 frange.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_harm 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 specan, dfts, ffts, dfDTW and fFDTW. Frequency measures can be made by the function or input by the user (see 'custom.contour' argument). If frange = TRUE the function uses frange.detec 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_freqs 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 (<marceloa27@gmail.com>)
trackfreqs

References


See Also

speccreator for creating spectrograms from selections, snrspecs for creating spectrograms to optimize noise margins used in sig2noise

Other spectrogram creators: color.spectro, dfDTW, dfts, ffDTW, ffts, multi_DTW, phylo_spectro, snrspecs, sp.en.ts, speccreator

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, img = FALSE, ssmooth = 600, wl = 300, flist = "Cryp.soui.wav",
                   path = tempdir())

  #track dominant frequency graphs with freq range detection
  trackfreqs(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 dfts
  df <- dfts(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 dfts output into trackfreqs
  trackfreqs(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
  trackfreqs(X = ad[!is.na(ad$start),], flim = c(0, 5), ovlp = 90, it = "tiff",
             bp = c(1, 3), contour = "both", wl = 300, path = tempdir())
}

track_harm

Track harmonic frequency contour

Description

track_harm tracks the frequency contour of the dominant harmonic.

Usage

track_harm(wave, f, wl = 512, wn = "hanning", ovlp = 0, fftw = FALSE, at = NULL, tlim = NULL, threshold = 10, bandpass = NULL, clip = NULL, plot = TRUE, xlab = "Times (s)", ylab = "Frequency (kHz)", ylim = c(0, f/2000), adjust.wl = FALSE, dfreq = FALSE, ...)

Arguments

wave A 'wave' object produced by readWave or similar functions.
f Sampling frequency of the wave object (in Hz). Does not need to be specified if embedded in wave.
wl A numeric vector of length 1 specifying the window length for the FFT, default is 512.
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).
ovlp Numeric vector of length 1 specifying % of overlap between two consecutive time windows, as in spectro. Default is 0.
fftw if TRUE calls the function FFT of the library fftw. See Notes of the spectro function. Default is FALSE.
at Time position where the harmonic frequency contour has to be computed (in seconds). Default is NULL.
tlim time range in which to measure frequency contours. Default is NULL (which means it will measure across the entire wave object).
threshold Amplitude threshold (%) for dominant frequency and detection. Default is 10.
bandpass A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz).
clip A numeric value to select dominant frequency values according to their amplitude in reference to a maximal value of 1 for the whole signal (has to be >0 & < 1).
plot Logical, if TRUE plots the dominant frequency against time. Default is TRUE.
xlab Label of the time axis.
ylab Label of the frequency axis.
try_na

A numeric vector of length 2 for the frequency limit of the spectrogram (in kHz), as in `spectro`. Default is c(0, f/2000).

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 FALSE.

Logical. If TRUE seeewave’s `dfreq` is used instead. Default is FALSE.

Additional arguments to be passed to the plotting function.

Details

This is a modified version of seewave’s `dfreq` function that allows to track the frequency contour of a dominant harmonic even when the highest amplitude jumps between harmonics. The arguments and default values of the original `dfreq` function have been kept unchanged to facilitate switching between the 2 functions.

Author(s)

Jerome Sueur, modified by Marcelo Araya-Salas (<marceloa27@gmail.com>)

References


See Also

`trackfreqs` for tracking frequencies iteratively on selections tables.

try_na

An R expression to try.

Logical to control whether the report of error messages is suppressed. Default is TRUE.

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.

Description

try_na silly wrapper for `try` function that returns an NA if an error is found. TO BE DEPRECATED IN FUTURE VERSIONS.

Usage

`try_na(expr, silent = TRUE, outFile)`

Arguments

expr

silent

outFile

Wrapper for "try" function

Description

try naive wrapper for `try` function that returns an NA if an error is found. TO BE DEPRECATED IN FUTURE VERSIONS.

Usage

`try_na(expr, silent = TRUE, outFile)`

Arguments

expr

silent

outFile

Wrapper for "try" function

Description

try naive wrapper for `try` function that returns an NA if an error is found. TO BE DEPRECATED IN FUTURE VERSIONS.

Usage

`try_na(expr, silent = TRUE, outFile)`

Arguments

expr

silent

outFile
Details

This is a silly wrapper on \texttt{try} that returns an ‘NA’ if any error occurs during the evaluation of a expression. See \texttt{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 (<marceloa27@gmail.com>)

References


Examples

\begin{verbatim}
{
  # 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))
}
\end{verbatim}
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

 querxc: Download recordings and/or metadata from 'Xeno-Canto'
 sim_songs: Simulate animal vocalizations

Managing sound files

 selection_table: Create 'selection_table' class objects
 mp32wav: Convert several .mp3 files in working directory to .wav format
 checksels: Check whether selections can be read by subsequent functions
 checkwavs: Check whether .wav files can be read by subsequent functions and the minimum windows length ("wl" argument) that can be used
 fixwavs: Fix .wav files so they can be read by other functions
 resample_est: Resample wave objects in extended selection tables
 wavdur: Determine the duration of sound files
 cut_sels: Cut selections from a selection table into individual sound files
 rm_sil: Remove silence segments from wave files
 rm_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

- **autodetec**: Automatically detect start and end of acoustic signals
- **manualoc**: Interactive spectrographic view to measure start and end of acoustic signals
- **autodetec**: Automatic detection of acoustic signals based on amplitude
- **seltailor**: Interactive view of spectrograms to tailor start and end of selections
- **sig2noise**: Measure signal-to-noise ratio across multiple files
- **trackfrequ**: Create spectrograms to visualize frequency measurements
- **filtersels**: Filter selection data frames based on filtered image files
- **frange**: Detect frequency range iteratively from signals in a selection table
- **frange.detec**: Detect frequency range in a Wave object
- **specan**: Measure acoustic parameters on selected acoustic signals
- **mfcc_stats**: Calculate descriptive statistics on Mel-frequency cepstral coefficients
- **xcorr**: Pairwise cross-correlation of multiple signals
- **dfts**: Extract the dominant frequency values across the signal as a time series
- **ffts**: Extract the fundamental frequency values across the signal as a time series
- **sp.en.ts**: Extract the spectral entropy values across the signal as a time series
- **dfDTW**: Calculate acoustic dissimilarity using dynamic time warping on dominant frequency contours
- **ffDTW**: Calculate acoustic dissimilarity using dynamic time warping on fundamental frequency contours
- **compare.methods**: Produce graphs to visually assess performance of acoustic distance measurements
- **coor.test**: Assess statistical significance of singing coordination
- **ovlp_sels**: Find selections that overlap in time within a given sound file
- **track_harm**: Track harmonic frequency contour

Graphical outputs

- **xcmaps**: 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
- **coor.graph**: Create graphs of coordinated singing
- **color.spectro**: Highlight spectrogram regions
- **lspec**: Produce spectrograms of whole recordings split into multiple rows
- **lspec2pdf**: Combine lspec images to single pdf files
- **specreator**: Create spectrograms of manualoc selections
- **snrspecs**: Create spectrograms to visualize margins over which noise will be measured by sig2noise
- **phylo_spectro**: Add spectrograms onto phylogenetic trees
warbleR_options

Author(s)
Marcelo Araya-Salas & Grace Smith Vidaurre
Maintainer: Marcelo Araya-Salas (<marceloa27@gmail.com>)

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.
warbleR_options

- **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 (<marceloa27@gmail.com>)

**Examples**

```r
{
# load data and save in 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"))

# 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)
}
```
**wavdur**  
*Measure the duration of sound files*

**Description**

`wavdur` measures the duration of sound files in `.wav` format.

**Usage**

```r
wavdur(files = NULL, path = NULL)
```

**Arguments**

- **files**: 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 `path`.
- **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 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 (<marceloa27@gmail.com>)

**References**


**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"))
  wavdur(path = tempdir())
}
```
Description

`wav_info` is a wrapper for `selection_table` that returns wave file information.

Usage

`wav_info(path = NULL, parallel = 1, pb = TRUE)`

Arguments

- `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

This function is a wrapper for `selection_table` that returns a data frame with the following descriptive parameters for each wave 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`: wave file size in MB
- `samples`: number of samples in the sound file

Value

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

Author(s)

Marcelo Araya-Salas (<marceloa27@gmail.com>)

References

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 .wav files, 2) “sel”: number of the selections, 3) “start”: start time of selections, 4) “end”: end time of selections. The output of manualoc or autodetec 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 (<marceloa27@gmail.com>)

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)
}
```

xcmaps

Maps of 'Xeno-Canto' recordings by species

Description

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

Usage

```r
xcmaps(X, img = TRUE, it = "jpeg", res = 100, labels = FALSE, path = NULL)
```
Arguments

- **X**: Data frame output from `querxc`.
- **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.

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 `querxc` as input. Maps can be displayed in the graphic device or saved as images in the working directory.

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 (<marceloa27@gmail.com>) and Grace Smith Vidaurre

References


Examples

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

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

## End(Not run)```
**xcorr**

*Spectrogram cross-correlation*

**Description**

`xcorr` estimates the similarity of two spectrograms by means of spectrographic cross-correlation.

**Usage**

```r
xcorr(X, wl = 512, bp = "pairwise.freq.range", ovlp = 90, dens = NULL, wn = "hanning", cor.method = "pearson", parallel = 1, path = NULL, pb = TRUE, na.rm = FALSE, cor.mat = TRUE, compare.matrix = NULL)
```

**Arguments**

- `X` '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).
- `wl` A numeric vector of length 1 specifying the window length of the spectrogram, default is 512.
- `bp` A numeric vector of length 2 for the lower and upper limits of a frequency bandpass filter (in kHz) or "pairwise.freq.range" (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.
- `ovlp` Numeric vector of length 1 specifying % of overlap between two consecutive windows, as in `spectro`. Default is 90. High values of ovlp slow down the function but produce more accurate results.
- `dens` DEPRECATED.
- `wn` A character vector of length 1 specifying the window name as in `ftwindow`.
- `cor.method` A character vector of length 1 specifying the correlation method as in `cor`.
- `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`.
- `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.
- `cor.mat` Logical. If `TRUE` only the correlation matrix is returned. Default is `TRUE`.
- `compare.matrix` A character matrix with 2 columns indicating the selections to be compared (column 1 vs column 2). The columns must contained 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 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.
Details

This function calculates the pairwise similarity of multiple signals by means of spectrogram cross-correlation. This method "slides" one spectrogram over the other 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. To accomplish this the margins of the signals are expanded by half the duration of the signal both before and after the provided time coordinates. 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 corr.mat is TRUE the function returns a matrix with the maximum (peak) correlation for each pairwise comparison. Otherwise it will return a list that includes 1) a data frame with the correlation statistic for each "sliding" step, 2) a matrix with the maximum correlation for each pairwise comparison, and 3) the frequency range.

Author(s)

Marcelo Araya-Salas <marceloa27@gmail.com>

References


See Also

mfcc_stats, specan, df_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
xcor <- xcorr(X = lbh_selec_table, wl = 300, ovlp = 90, path = tempdir())

# 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
xcorr <- xcorr(X = lbh_selec_table, compare.matrix = cmp.mt,
               wl = 300, ovlp = 90, path = tempdir())
}
Index

*Topic datasets
  lbh_selec_table, 54
  new_function_names, 68
  selec_table, 82
  sim.coor.sing, 91
  sim_coor_sing, 91

acoustat, 104, 105
analyze, 104, 105
approx, 33, 35, 38, 40, 67, 101
auto_detec, 112
autodetect, 3, 13, 16, 20, 30, 31, 34, 37, 39, 41, 45, 48, 50, 62, 70, 81, 83, 86, 90, 97, 101, 103, 104, 106, 113, 122, 127
dtw, 22, 32, 37, 67
dtwDist, 31, 36, 66

FF, 40, 103, 116
ffDTW, 19, 20, 33, 36, 39, 40, 68, 73, 96, 102, 109, 116, 117, 122
ffts, 19, 22, 33, 36, 39, 68, 73, 96, 102, 109, 116, 117, 122

file.copy, 24
filtersels, 41, 64, 71, 87, 122
fix_extended_selection_table, 44, 121
fix_wavs, 65, 79
fixwavs, 14, 16, 25, 43, 63, 80, 81, 84, 121, 127
fppeaks, 104
frange, 45, 50, 122
frange.detec, 32, 35, 47, 48, 48, 116, 122
ftwindow, 8, 17, 32, 34, 37, 39, 46, 49, 59, 86, 94, 101, 104, 107, 110, 114, 118, 124, 130
fund, 40, 103, 104, 116
getPitchAutocor, 105
grey, 1, 5, 8, 18, 46, 49, 56, 60, 87, 108, 116
grey, 2, 5, 8, 18, 46, 47, 49, 56, 60, 87, 108, 116
grey3, 5, 8, 18, 46, 49, 56, 60, 87, 108, 116

H, 105

inflections, 50
isExtended_selection_table, 52
isSelected_table, 52, 53

ladderize, 72
lbh_selec_table, 30, 54, 71
lspec, 4, 41, 55, 57, 58, 122
lspec2pdf, 41, 57, 57, 122

manual_loc, 89, 94
manualoc, 6, 13, 16, 20, 30, 31, 34, 37, 39, 41, 45, 59, 62, 70, 83, 86, 88, 96, 97, 100, 101, 103, 104, 106, 113, 122, 127
meanspec, 46–49, 105, 118
melfcc, 62
mfcc_stats, 20, 61, 122, 128, 131
move.imgs, 63, 69, 113
mp32wav, 65, 79, 121
multi_DTW, 19, 33, 36, 38, 40, 66, 73, 96, 102, 109, 117
new_function_names, 68
normalize, 65, 66
object.size, 84
open_wd, 64, 69, 113
ovlp_sels, 70, 122
par, 9, 72, 95, 107, 114
pdf, 12
phylo_spectro, 19, 33, 36, 38, 40, 68, 71, 96, 102, 109, 117, 122
plot.phylo, 72, 73
prcomp, 21, 22
querxc, 74, 93, 121, 129
rainbow, 1, 5, 8, 18, 46, 49, 56, 60, 87, 108, 116
read_wave, 76
readMP3, 66
readWave, 14, 17, 48, 76, 77, 118
resample, 65
resample_est, 78, 121
rm_channels, 79, 121
rm_sil, 63, 80, 80, 121
scale, 21, 32, 37, 67
select.table, 82
selection_table, 14, 52–54, 72, 77, 78, 83, 107, 121, 126, 127
seltailor, 16, 30, 61, 86, 122
set.seed, 92
sfm, 105
sh, 105
sig2noise, 19, 33, 36, 38, 40, 68, 89, 94, 95, 102, 109, 117, 122
sim.coor.sing, 91
sim.coor.sing, 91
sim_songs, 92, 121
snrspecs, 19, 33, 36, 38, 40, 68, 73, 90, 94, 102, 109, 117, 122
song_param, 56, 97
sort_coims, 99
sp.en.ts, 19, 33, 36, 38, 40, 68, 73, 96, 100, 109, 117, 122
spec_param, 109, 110
specan, 16, 19, 20, 22, 84, 96, 98, 103, 109, 116, 122, 131
specprop, 104, 105
speccreator, 5, 19, 21, 33, 36, 38, 40, 41, 68, 72, 73, 96, 102, 106, 117, 122
split_wavs, 64, 69, 112
th, 105
timer, 6
topo.1, 5, 8, 18, 46, 49, 56, 60, 87, 108, 116
track_harm, 35, 116, 118, 122
trackfrequs, 19, 33, 35–38, 40, 51, 67, 68, 73, 96, 101, 102, 109, 113, 119, 122
try, 119, 120
try_na, 119
warbleR, 63, 64, 68, 69, 83, 120
warbleR-package (warbleR), 120
warbleR_options, 24, 41, 69, 89, 90, 123
wav_info, 126
wavdur, 121, 125
weighted.mean, 98
wpd_features, 127
writeWave, 30, 66
xcmaps, 75, 76, 122, 128
xcorr, 20, 22, 84, 122, 130