Package 'voice'

May 8, 2026

Type Package

Title Speaker Recognition, Voice Analysis and Mood Inference via Music Theory

Version 0.5.4

Date 2025-07-14

Maintainer Zabala Filipe J. <filipezabala@gmail.com>

Description Provides tools for audio data analysis, including feature extraction, pitch detection, and speaker identification. Designed for voice research and signal processing applications.

License GPL-3

Depends R (>= 4.0.0)

Imports arrangements, dplyr, ggplot2, R.utils, reticulate, seewave, tabr, tibble, tidyselect, tuneR, wrassp, zoo, htmltools, httr

Suggests gm, knitr, tidyverse

URL https://github.com/filipezabala/voice

BugReports https://github.com/filipezabala/voice/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Zabala Filipe J. [cre, aut]

Repository CRAN

Date/Publication 2025-07-14 19:50:02 UTC

Contents

`assign_notes` *Assign musical notes*

Description

Assign musical notes in Scientific Pitch Notation or other variant. See `voice::notes()`. The notes are cut considering f0 to ensure alignment.

Usage

```
assign_notes(
  x,
  fmt = 0,
  min_points = 4,
  min_percentile = 0.75,
  max_na_prop = 1
)
```

Arguments

`x`	Media dataset from voice::extract_features().
`fmt`	Either F0 or formant frequency (in Hz). Default: `fmt = 0`.
`min_points`	Minimum number of points for audio section. Default: `min_points = 4`.
`min_percentile`	Minimum percentile value of gain to be included on the average of `fmt`. Default: `min_percentile = 0.75`.
`max_na_prop`	Maximum proportion os NAs on gain sector. Default: `max_na_prop = 1`.

Examples

```
library(voice)
# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)
# Media dataset
M <- extract_features(path2wav)
assign_notes(M, fmt = 0) # f0
assign_notes(M, fmt = 1) # f1
assign_notes(M, fmt = 2) # f2
```

audio_time *Returns the total time of audio files in seconds*

Description

Returns the total time of audio files in seconds

Usage

```
audio_time(x, filesRange = NULL, recursive = FALSE)
```

Arguments

x	Either a WAV file or a directory containing WAV files.
filesRange	The desired range of directory files (default: NULL, i.e., all files).
recursive	Logical. Should the listing recursively into directories? (default: FALSE) Used by base::list.files.

Value

A tibble containing file name <chr> and audio time <dbl> in seconds.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

# Tibble containing file name and audio time
(at <- voice::audio_time(unique(dirname(path2wav))))
str(at)
```

---

check_chords *Check chords*

Description

Check the sequence of musical notes for chords.

Usage

```
check_chords(x, window = 3, try_perm = FALSE)
```

Arguments

x	A vector containing a sequence of musical notes.
window	Size of window of notes to be checked. Default: 3.
try_perm	Logical. Must try all notes permutations of notes? Default: FALSE.

Examples

```
## Not run:
library(voice)
check_chords(c('C','E','G'), window = 3, try_perm = FALSE)
check_chords(c('C','E','G'), window = 3, try_perm = TRUE)
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)
M <- extract_features(path2wav)
M$gain[is.na(M$f0)] <- NA
# assigning notes
f0_spn <- assign_notes(M, fmt = 0)
check_chords(f0_spn, window = 3, try_perm = FALSE)
check_chords(f0_spn, window = 3, try_perm = TRUE)
check_chords(f0_spn, window = 4, try_perm = TRUE)

## End(Not run)
```

---

cut_audio *Cut audio vectors*

Description

Cut vectors

Usage

```
cut_audio(x, byvar = x)
```

Arguments

x	A vector containing the feature to be cut by byvar.
byvar	A vector containing the variable to cut by.

Examples

```
library(voice)
# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)
# Media dataset
M <- extract_features(path2wav)
cut_audio(M$f0)
cut_audio(M$gain, M$f0)
```

`diarize` *Who spoke when?*

Description

Diarization of WAV audios.

Usage

```
diarize(
  fromWav,
  toRttm = NULL,
  autoDir = FALSE,
  pycall = "~/miniconda3/envs/pyvoice/bin/python",
  token = NULL
)
```

Arguments

`fromWav`	Either a file or a directory containing WAV files.
`toRttm`	A directory to write RTTM files. If the default `toRttm = NULL` is used, `'./voiceAudios/rttm'` is created and used.
`autoDir`	Logical. Must the directories tree be created? Default: `FALSE`. See 'Details'.
`pycall`	Python call. See https://github.com/filipezabala/voice for details.
`token`	Access token needed to instantiate pretrained speaker diarization pipeline from pyannote.audio. #1 Install pyannote.audio 3.1 with pip install pyannote.audio (already listed @ https://raw.githubusercontent.com/filipezabala/voice/master/requirements.txt). #2. Accept https://hf.co/pyannote/segmentation-3.0 user conditions. #3 Accept https://huggingface.co/pyannote/speaker-diarization-3.1 user conditions. #4. Create access token.

Details

When `autoDir = TRUE`, the following directories are created: `'../mp3'`,`'../rttm'`, `'../split'` and `'../musicxml'`. Use `getwd()` to find the parent directory `'../'`.

Value

RTTM files in NIST standard. See 'voice::read_rttm'.

Examples

```
## Not run:
library(voice)

wavDir <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)
```

```
voice::diarize(fromWav = unique(dirname(wavDir)),
toRttm = tempdir(),
token = NULL) # Must enter a token! See documentation.

(rttm <- dir(tempdir(), '.[Rr][Tt][Tt][Mm]$', full.names = TRUE))
file.info(rttm)

## End(Not run)
```

`duration` *Duration of sequences*

Description

Duration of sequences

Usage

```
duration(x, windowShift = 5)
```

Arguments

`x`	A vector containing symbols and `NA`.
`windowShift`	Window shift to duration in ms (default: 5.0).

Value

A data frame with duration in number of lines/ocurrences (`dur_line`), milliseconds (`dur_ms`) and proportional (`dur_prop`).

Examples

```
library(voice)
duration(letters)
duration(c('a','a','a',letters,'z'))

nts <- c('NA','NA','A3','A3','A3','A3','A#3','B3','B3','C4','C4','C4','C4',
'C4','C4','C#4','C4','C4','C4','B3','A#3','NA','NA','NA','NA','NA','NA','NA',
'NA','NA','NA','NA','NA','NA','NA','NA','NA','NA','NA','D4','D4','D4','C#4',
'C#4','C#4','C4','C4','B3','B3','A#3','A#3','A3','A3','G3','G#3','G3','F#3')
duration(nts)
```

embed_audio	*These functions are sourced from the 'embedr' package by Michael McCarthy, under MIT License: https://github.com/mccarthy-m-g/embedr/blob/master/LICENSE.md This inclusion is temporary and will be discontinued once 'embedr' is available on CRAN. See https://github.com/mccarthy-m-g/embedr for more details.*

Description

Embed audio in R Markdown documents

Usage

```
embed_audio(
  src,
  type = c("mpeg", "ogg", "wav"),
  attribute = c("controls", "autoplay", "loop", "muted", "preload", "none"),
  id = "",
  placeholder = ""
)
```

Arguments

src	A path or URL to the media file.
type	The type of media file specified in 'src'.
attribute	A character vector specifying which attributes to use. "none" can be used if no attributes are desired.
id	A character string specifying a unique ID for the element. Can be used by CSS or JavaScript to perform certain tasks for the element with the specific ID.
placeholder	The placeholder text to use when the output format is not HTML.

Details

'embed_audio()' provides a standard way to embed audio in R Markdown documents when the output format is HTML, and to print placeholder text when the output format is not HTML.

'embed_audio()' is a wrapper for the HTML5 '<audio>' element that prints HTML '<audio>' code in HTML documents built by R Markdown and placeholder text in non-HTML documents built by R Markdown. This function may be useful for conditional output that depends on the output format. For example, you may embed audio in an R Markdown document when the output format is HTML, and print placeholder text when the output format is LaTeX.

The function determines output format using [knitr::is_html_output()]. By default, these formats are considered as HTML formats: 'c('markdown', 'epub', 'html', 'html5', 'revealjs', 's5', 'slideous', 'slidy')'.

Value

If ‘knitr::is_html_output()‘ is ‘TRUE‘, returns HTML ‘<audio>‘ code. If ‘knitr::is_html_output()‘ is ‘FALSE‘, returns placeholder text.

Note

This function is supposed to be used in R code chunks or inline R code expressions. You are recommended to use forward slashes (/) as path separators instead of backslashes in the file paths.

Examples

```
# By default, embed_audio() embeds an audio element with playback controls
embed_audio(mp3)

# To change the attributes of the audio element, use `attribute`
embed_audio(mp3, attribute = c("controls", "loop"))

# To add placeholder text for non-HTML documents, use `placeholder`
embed_audio(mp3, placeholder = "This is placeholder text.")

## Not run:
# embed_audio() is intended to be used in R Markdown code chunks or inline
# expressions. The following creates and knits an R Markdown document to
# HTML and PDF in your current working directory for you to inspect:
library(rmarkdown)
writeLines(c("# Hello embedr!",
"```{r embed-audio, echo=TRUE}",
"embed_audio(mp3, placeholder = 'This is placeholder text.')",
"```"), "test.Rmd")
render("test.Rmd", output_format = c('html_document', 'pdf_document'))

# Delete test files created by example code
unlink(c("test.Rmd", "test.html", "test.pdf"))

## End(Not run)
```

---

embed_video *Embed video in R Markdown documents*

Description

‘embed_video()‘ provides a standard way to embed video in R Markdown documents when the output format is HTML, and to print placeholder text when the output format is not HTML.

Usage

```
embed_video(
  src,
  type = c("mp4", "webm", "ogg"),
```

```
  width = "320",
  height = "240",
  attribute = c("controls", "autoplay", "loop", "muted", "preload", "none"),
  thumbnail = NULL,
  id = "",
  placeholder = ""
)
```

Arguments

src	A path or URL to the media file.
type	The type of media file specified in 'src'.
width	The width of the video, in pixels.
height	The height of the video, in pixels.
attribute	A character vector specifying which attributes to use. "none" can be used if no attributes are desired.
thumbnail	A path to an image.
id	A character string specifying a unique ID for the element. Can be used by CSS or JavaScript to perform certain tasks for the element with the specific ID.
placeholder	The placeholder text to use when the output format is not HTML.

Details

'embed_video()' is a wrapper for the HTML5 '<video>' element that prints HTML '<video>' code in HTML documents built by R Markdown and placeholder text in non-HTML documents built by R Markdown. This function may be useful for conditional output that depends on the output format. For example, you may embed video in an R Markdown document when the output format is HTML, and print placeholder text when the output format is LaTeX.

The function determines output format using [knitr::is_html_output()]. By default, these formats are considered as HTML formats: 'c('markdown', 'epub', 'html', 'html5', 'revealjs', 's5', 'slideous', 'slidy')'.

Value

If 'knitr::is_html_output()' is 'TRUE', returns HTML '<video>' code. If 'knitr::is_html_output()' is 'FALSE', returns placeholder text.

Note

This function is supposed to be used in R code chunks or inline R code expressions. You are recommended to use forward slashes (/) as path separators instead of backslashes in the file paths.

Examples

```
# By default, embed_video() embeds a video element with playback controls
embed_video(mp4)
```

```
# To change the attributes of the video element, use `attribute`
embed_video(mp4, attribute = c("controls", "loop"))

# To add a thumbnail to the video element, use `thumbnail`
embed_video(mp4, thumbnail = png)

# To add placeholder text for non-HTML documents, use `placeholder`
embed_video(mp4, placeholder = "This is placeholder text.")

## Not run:
# embed_video() is intended to be used in R Markdown code chunks or inline
# expressions. The following creates and knits an R Markdown document to
# HTML and PDF in your current working directory for you to inspect:
library(rmarkdown)
writeLines(c("# Hello embedr!",
"```{r embed-video, echo=TRUE}",
"embed_video(mp4, thumbnail = png, placeholder = 'This is placeholder text.')",
"```"), "test.Rmd")
render("test.Rmd", output_format = c('html_document', 'pdf_document'))

# Delete test files created by example code
unlink(c("test.Rmd", "test.html", "test.pdf"))

## End(Not run)
```

`enrich_rttm` *Enrich RTTM files*

Description

Enrich Rich Transcription Time Marked (RTTM) files obtained from '`voice::read_rttm`'.

Usage

```
enrich_rttm(listRttm, silence.gap = 0.5, as.tibble = TRUE)
```

Arguments

`listRttm`	A list containing RTTM files.
`silence.gap`	The silence gap (in seconds) between adjacent words in a keyword. Rows with `tdur` <= `silence.gap` are removed. (default: `0.5`)
`as.tibble`	Logical. Should it return a tibble?

Value

A list containing either data frames or tibbles obtained from standard RTTM files. See 'voice::read_rttm'.

References

https://www.nist.gov/system/files/documents/itl/iad/mig/KWS15-evalplan-v05.pdf

See Also

`voice::read_rttm`

Examples

```
library(voice)

url0 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock0.rttm'
destfile0 <- paste0(tempdir(), '/sherlock0.rttm')
download.file(url0, destfile = destfile0)
url1 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock1.rttm'
destfile1 <- paste0(tempdir(), '/sherlock1.rttm')
download.file(url0, destfile = destfile1)

rttm <- voice::read_rttm(dirname(destfile0))
(er <- voice::enrich_rttm(rttm))
class(er)
lapply(er, class)
```

---

expand_model *Expand model*

Description

Expand model given `y` and `x` variables.

Usage

```
expand_model(y, x, k)
```

Arguments

`y`	The Y variable.
`x`	The X variables.
`k`	Number of additive components.

Value

A `char` vector containing the expanded models.

Examples

```
library(voice)

expand_model('y', LETTERS[1:4], 1)
expand_model('y', LETTERS[1:4], 2)
expand_model('y', LETTERS[1:4], 3)
```

```
expand_model('y', LETTERS[1:4], 4)

# multiple models using apply functions
nx <- 10 # number of X variables to be used
models <- lapply(1:nx, expand_model, y = 'y', x = LETTERS[1:nx])
names(models) <- 1:nx
models
sum(sapply(models, length)) # total of models
```

`extract_features` *Extract audio features*

Description

Extracts features from WAV audio files.

Usage

```
extract_features(
  x,
  features = c("f0", "fmt", "gain"),
  filesRange = NULL,
  sex = "u",
  windowShift = 10,
  numFormants = 8,
  numcep = 12,
  dcttype = c("t2", "t1", "t3", "t4"),
  fbtype = c("mel", "htkmel", "fcmel", "bark"),
  resolution = 40,
  usecmp = FALSE,
  mc.cores = 1,
  full.names = TRUE,
  recursive = FALSE,
  check.mono = FALSE,
  stereo2mono = FALSE,
  overwrite = FALSE,
  freq = 44100,
  round.to = NULL,
  verbose = FALSE,
  pycall = "~/miniconda3/envs/pyvoice/bin/python"
)
```

Arguments

x	A vector containing either files or directories of audio files in WAV format.
features	Vector of features to be extracted. (Default: `'f0','fmt','gain'`). Available features: `'f0','f0_mhs','f0_praat','fmt','fmt_praat','zcr','rms','gain','rfc','ac','ce`

filesRange	The desired range of directory files (Default: NULL, i.e., all files). Should only be used when all the WAV files are in the same folder.
sex	= <code> set sex specific parameters where <code> = 'f'[emale], 'm'[ale] or 'u'[nknown] (Default: 'u'). Used as 'gender' by wrassp::ksvF0, wrassp::forest and wrassp::mhsF0.
windowShift	= <dur> set analysis window shift to <dur>ation in ms (Default: 5.0). Used by wrassp::ksvF0, wrassp::forest, wrassp::mhsF0, wrassp::zcrana, wrassp::rfcana, wrassp::acfana, wrassp::cepstrum, wrassp::dftSpectrum, wrassp::cssSpectrum and wrassp::lpsSpectrum.
numFormants	= <num> <num>ber of formants (Default: 8). Used by wrassp::forest.
numcep	Number of Mel-frequency cepstral coefficients (cepstra) to return (Default: 12). Used by tuneR::melfcc.
dcttype	Type of DCT used. 't1' or 't2', 't3' for HTK 't4' for feacalc (Default: 't2'). Used by tuneR::melfcc.
fbtype	Auditory frequency scale to use: 'mel', 'bark', 'htkmel', 'fcmel' (Default: 'mel'). Used by tuneR::melfcc.
resolution	= <freq> set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (Default: 40.0). Used by wrassp::cssSpectrum, wrassp::dftSpectrum and wrassp::lpsSpectrum.
usecmp	Logical. Apply equal-loudness weighting and cube-root compression (PLP instead of LPC) (Default: FALSE). Used by tuneR::melfcc.
mc.cores	Number of cores to be used in parallel processing. (Default: 1)
full.names	Logical. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned. (Default: TRUE) Used by base::list.files.
recursive	Logical. Should the listing recursively into directories? (Default: FALSE) Used by base::list.files.
check.mono	Logical. Check if the WAV file is mono. (Default: TRUE)
stereo2mono	(Experimental) Logical. Should files be converted from stereo to mono? (Default: TRUE)
overwrite	(Experimental) Logical. Should converted files be overwritten? If not, the file gets the suffix _mono. (Default: FALSE)
freq	Frequency in Hz to write the converted files when stereo2mono=TRUE. (Default: 44100)
round.to	Number of decimal places to round to. (Default: NULL)
verbose	Logical. Should the running status be showed? (Default: FALSE)
pycall	Python call. See https://github.com/filipezabala/voice for details.

Details

The feature 'df' corresponds to 'formant dispersion' (df2:df8) by Fitch (1997), 'pf' to formant position' (pf1:pf8) by Puts, Apicella & Cárdena (2011), 'rf' to 'formant removal' (rf1:rf8) by Zabala (2023), 'rcf' to 'formant cumulated removal' (rcf2:rcf8) by Zabala (2023) and 'rpf' to 'formant position removal' (rpf2:rpf8) by Zabala (2023). The 'fmt_praat' feature may take long time processing. The following features may contain a variable number of columns: 'cep', 'dft', 'css' and 'lps'.

Value

A Media data frame containing the selected features.

References

Levinson N. (1946). The Wiener (root mean square) error criterion in filter design and prediction. Journal of Mathematics and Physics, 25(1-4), 261–278. (doi:10.1002/SAPM1946251261)

Durbin J. (1960). "The fitting of time-series models." Revue de l'Institut International de Statistique, pp. 233–244. (https://www.jstor.org/stable/1401322)

Cooley J.W., Tukey J.W. (1965). "An algorithm for the machine calculation of complex Fourier series." Mathematics of computation, 19(90), 297–301. (https://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/)

Wasson D., Donaldson R. (1975). "Speech amplitude and zero crossings for automated identification of human speakers." IEEE Transactions on Acoustics, Speech, and Signal Processing, 23(4), 390–392. (https://ieeexplore.ieee.org/document/1162690)

Allen J. (1977). "Short term spectral analysis, synthesis, and modification by discrete Fourier transform." IEEE Transactions on Acoustics, Speech, and Signal Processing, 25(3), 235– 238. (https://ieeexplore.ieee.org/document/1162950)

Schäfer-Vincent K. (1982). "Significant points: Pitch period detection as a problem of segmentation." Phonetica, 39(4-5), 241–253. (doi:10.1159/000261665)

Schäfer-Vincent K. (1983). "Pitch period detection and chaining: Method and evaluation." Phonetica, 40(3), 177–202. (doi:10.1159/000261691)

Ephraim Y., Malah D. (1984). "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator." IEEE Transactions on acoustics, speech, and signal processing, 32(6), 1109–1121. (https://ieeexplore.ieee.org/document/1164453)

Delsarte P., Genin Y. (1986). "The split Levinson algorithm." IEEE transactions on acoustics, speech, and signal processing, 34(3), 470–478. (https://ieeexplore.ieee.org/document/1164830)

Jackson J.C. (1995). "The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice." Technical report, Carnegie-Mellon University Pittsburgh PA Schoo; of Computer Science.

Fitch, W.T. (1997) "Vocal tract length and formant frequency dispersion correlate with body size in rhesus macaques." J. Acoust. Soc. Am. 102, 1213 – 1222. (doi:10.1121/1.421048)

Boersma P., van Heuven V. (2001). Praat, a system for doing phonetics by computer. Glot. Int., 5(9/10), 341–347. (https://www.fon.hum.uva.nl/paul/papers/speakUnspeakPraat_glot2001.pdf)

Ellis DPW (2005). "PLP and RASTA (and MFCC, and inversion) in Matlab." Online web resource.

Puts, D.A., Apicella, C.L., Cardenas, R.A. (2012) "Masculine voices signal men's threat potential in forager and industrial societies." Proc. R. Soc. B Biol. Sci. 279, 601–609. (doi:10.1098/rspb.2011.0829)

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

# minimal usage
M1 <- extract_features(path2wav)
M2 <- extract_features(dirname(path2wav))
identical(M1,M2)
table(basename(M1$wav_path))

# limiting filesRange
M3 <- extract_features(path2wav, filesRange = 3:6)
table(basename(M3$wav_path))
```

---

feat_summary *Features summary*

Description

Returns summary measures of 'voice::extract_features'.

Usage

```
feat_summary(
  x,
  groupBy = "wav_path",
  wavPath = unique(x$wav_path),
  wavPathName = "wav_path",
  features = "f0",
  filesRange = NULL,
  sex = "u",
  windowShift = 10,
  numFormants = 8,
  numcep = 12,
  dcttype = c("t2", "t1", "t3", "t4"),
  fbtype = c("mel", "htkmel", "fcmel", "bark"),
  resolution = 40,
  usecmp = FALSE,
  mc.cores = 1,
  full.names = TRUE,
  recursive = FALSE,
  check.mono = FALSE,
  stereo2mono = FALSE,
  overwrite = FALSE,
  freq = 44100,
```

```
  round.to = 4,
  verbose = FALSE
)
```

Arguments

x	An Extended data frame to be tagged with media information.
groupBy	A variable to group the summary measures. The argument must be a character vector. (Default: groupBy = 'wav_path').
wavPath	A vector containing the path(s) to WAV files. May be both as dirname or basename formats.
wavPathName	A string containing the WAV path name. (Default: wavPathName = 'wav_path').
features	Vector of features to be extracted. (Default: 'f0').
filesRange	The desired range of directory files (default: NULL, i.e., all files). Should only be used when all the WAV files are in the same folder.
sex	= <code> set sex specific parameters where <code> = 'f'[emale], 'm'[ale] or 'u'[nknown] (Default: 'u'). Used as 'gender' by wrassp::ksvF0, wrassp::forest and wrassp::mhsF0.
windowShift	= <dur> set analysis window shift to <dur>ation in ms (Default: 5.0). Used by wrassp::ksvF0, wrassp::forest, wrassp::mhsF0, wrassp::zcrana, wrassp::rfcana, wrassp::acfana, wrassp::cepstrum, wrassp::dftSpectrum, wrassp::cssSpectrum and wrassp::lpsSpectrum.
numFormants	= <num> <num>ber of formants (Default: 8). Used by wrassp::forest.
numcep	Number of Mel-frequency cepstral coefficients (cepstra) to return (Default: 12). Used by tuneR::melfcc.
dcttype	Type of DCT used. 't1' or 't2', 't3' for HTK 't4' for feacalc (Default: 't2'). Used by tuneR::melfcc.
fbtype	Auditory frequency scale to use: 'mel', 'bark', 'htkmel', 'fcmel' (Default: 'mel'). Used by tuneR::melfcc.
resolution	= <freq> set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (Default: 40.0). Used by wrassp::cssSpectrum, wrassp::dftSpectrum and wrassp::lpsSpectrum.
usecmp	Logical. Apply equal-loudness weighting and cube-root compression (PLP instead of LPC) (Default: FALSE). Used by tuneR::melfcc.
mc.cores	Number of cores to be used in parallel processing. (Default: 1)
full.names	Logical. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned. (Default: TRUE). Used by base::list.files.
recursive	Logical. Should the listing recursively into directories? (Default: FALSE) Used by base::list.files.
check.mono	Logical. Check if the WAV file is mono. (Default: TRUE)
stereo2mono	(Experimental) Logical. Should files be converted from stereo to mono? (Default: TRUE)

overwrite	(Experimental) Logical. Should converted files be overwritten? If not, the file gets the suffix _mono. (Default: FALSE)
freq	Frequency in Hz to write the converted files when stereo2mono=TRUE. (Default: 44100)
round.to	Number of decimal places to round to. (Default: NULL)
verbose	Logical. Should the running status be showed? (Default: FALSE)

Details

filesRange should only be used when all the WAV files are in the same folder.

Value

A tibble data frame containing summarized numeric columns using (1) mean, (2) standard deviation, (3) variation coefficient, (4) median, (5) interquartile range and (6) median absolute deviation.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

# creating Extended synthetic data
E <- dplyr::tibble(subject_id = c(1,1,1,2,2,2,3,3,3),
wav_path = path2wav)

# minimal usage
feat_summary(E)

# canonical data
feat_summary(E, groupBy = 'subject_id')
```

---

get_bit *Get bit rate*

Description

Get bit rate from WAV file.

Usage

```
get_bit(x)
```

Arguments

x	Wave object from ‘tuneR::readWave‘.

Value

Integer indicating the bit rate from a WAV file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

rw <- tuneR::readWave(path2wav[1])
voice::get_bit(rw)

rwl <- lapply(path2wav, tuneR::readWave)
sapply(rwl, voice::get_bit)
```

---

get_dur *Time duration*

Description

Get time duration from WAV file.

Usage

```
get_dur(x)
```

Arguments

x	Wave object from 'tuneR::readWave'.

Value

Numeric indicating the time duration in seconds from a WAV file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

rw <- tuneR::readWave(path2wav[1])
voice::get_dur(rw)

rwl <- lapply(path2wav, tuneR::readWave)
sapply(rwl, voice::get_dur)
```

get_left *Get left channel*

Description

Get left channel from WAV file.

Usage

```
get_left(x)
```

Arguments

x	Wave object from ‘tuneR::readWave‘.

Value

Numeric vector indicating the left channel from a WAV file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

rw <- tuneR::readWave(path2wav[1])
l <- voice::get_left(rw)
head(l)
length(l)
```

get_right *Get right channel*

Description

Get right channel from WAV file.

Usage

```
get_right(x)
```

Arguments

x	Wave object from ‘tuneR::readWave‘.

Value

Numeric vector indicating the right channel from a WAV file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

rw <- tuneR::readWave(path2wav[1])
r <- voice::get_right(rw)
head(r)
length(r)
```

---

get_samp.rate *Get sample rate*

Description

Get sample rate from WAV file.

Usage

```
get_samp.rate(x)
```

Arguments

x	Wave object from 'tuneR::readWave'.

Value

Integer indicating the sample rate from a WAV file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

rw <- tuneR::readWave(path2wav[1])
voice::get_samp.rate(rw)

rwl <- lapply(path2wav, tuneR::readWave)
sapply(rwl, voice::get_samp.rate)
```

get_tbeg *Time beginning*

Description

Get time beginning from a data frame in RTTM standard.

Usage

```
get_tbeg(x)
```

Arguments

x	A data frame in RTTM standard. See 'voice::read_rttm'.

Value

Numeric vector containing the time beginning in seconds.

Examples

```
library(voice)

url0 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock0.rttm'
download.file(url0, destfile = paste0(tempdir(), '/sherlock0.rttm'))

rttm <- voice::read_rttm(tempdir())
(gtb <- voice::get_tbeg(rttm$sherlock0.rttm))
class(gtb)
```

get_tdur *Time duration*

Description

Get time duration from a data frame in RTTM standard.

Usage

```
get_tdur(x)
```

Arguments

x	A data frame in RTTM standard. See 'voice::read_rttm'.

Value

Numeric vector containing the time duration in seconds.

Examples

```
library(voice)

url0 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock0.rttm'
download.file(url0, destfile = paste0(tempdir(), '/sherlock0.rttm'))

rttm <- voice::read_rttm(tempdir())
(gtd <- voice::get_tdur(rttm$sherlock0.rttm))
class(gtd)
```

`interp` *Interpolate vectors*

Description

Interpolate vactors, compressing to `compact.to` fraction. May remove zeros.

Usage

```
interp(
  y,
  compact.to,
  drop.zeros = FALSE,
  to.data.frame = FALSE,
  round.off = NULL,
  weight = NULL
)
```

Arguments

`y`	A vector or time series.
`compact.to`	Proportion of remaining points after compaction, between (including) 0 and 1. If equals to 1 and keep.zeros = TRUE, the original vector is presented.
`drop.zeros`	Logical. Drop repeated zeros? Default: `FALSE`.
`to.data.frame`	Logical. Convert to data frame? Default: `FALSE`.
`round.off`	Number of decimal places of the interpolated `y` Default: `NULL`.
`weight`	Vector of weights with same length of `y`. Default: `NULL`.

Value

A list of interpolated `x` and `y` values with length near to `compact.to*length(y)`.

See Also

`rm0`, `interp_mc`, `interp_df`

Examples

```
library(voice)

v1 <- 1:100
(c1 <- interp(v1, compact.to = 0.2))
length(c1$y)
plot(1:100, type = 'l')
points(c1$x, c1$y, col='red')

# with weight
(c2 <- interp(v1, compact.to = 0.2, weight = rev(v1)))
plot(c1$y)
points(c2$y, col = 'red')

(v2 <- c(1:5, rep(0,10), 1:10, rep(0,5), 10:20, rep(0,10)))
length(v2)
interp(v2, 0.1, drop.zeros = TRUE, to.data.frame = FALSE)
interp(v2, 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
interp(v2, 0.2, drop.zeros = TRUE)
interp(v2, 0.2, drop.zeros = FALSE)

(v3 <- c(rep(0,10), 1:20, rep(0,3)))
(c3 <- interp(v3, 1/3, drop.zeros = FALSE, to.data.frame = FALSE))
lapply(c3, length)
plot(v3, type = 'l')
points(c3$x, c3$y, col = 'red')

(v4 <- c(rnorm(1:100)))
(c4 <- interp(v4, 1/4, round.off = 3))
```

---

interp_df *Inperpolate data frames*

Description

Interpolate data frames using multicore, compressing to `compact.to` fraction. May remove zeros.

Usage

```
interp_df(
  x,
  compact.to,
  id = colnames(x)[1],
  colnum = NULL,
  drop.x = TRUE,
  drop.zeros = FALSE,
  to.data.frame = TRUE,
  round.off = NULL,
  weight = NULL,
```

```
  mc.cores = 1
)
```

Arguments

x	A data frame.
compact.to	Proportion of remaining points after interpolation. If equals to 1 and keep.zeros = TRUE, the original vector is presented.
id	The identification column. Default: colname of the first column of x.
colnum	A char vector indicating the numeric colnames. If NULL, uses the columns of the numeric class.
drop.x	Logical. Drop columns containing .x? Default: TRUE.
drop.zeros	Logical. Drop repeated zeros or keep 1 zero per null set? Default: FALSE.
to.data.frame	Logical. Should return a data frame? If FALSE returns a list. Default: TRUE.
round.off	Number of decimal places of the interpolated y. Default: NULL.
weight	Vector of weights with same length of y. Default: NULL.
mc.cores	The number of cores to mclapply. Default: 1.

Value

A data frame of interpolated values with nrow near to compact.to*length(x).

See Also

interp, interp_mc

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

# getting Media data frame via lean call
M <- extract_features(dirname(path2wav), features = c('f0','fmt'),
mc.cores = 1, verbose = FALSE)


(cM.df <- interp_df(M[,-(1:2)], 0.1, mc.cores = 1))
(cM.df2 <- interp_df(M[,-(1:2)], 0.1, drop.x = FALSE, mc.cores = 1))

dim(M)
dim(cM.df)
dim(cM.df2)
(cM.list <- interp_df(M[,-(1:2)], 0.1, to.data.frame = FALSE, mc.cores = 1))
```

interp_mc *Interpolate vectors using multicore*

Description

Interpolate vectors using multicore

Usage

```
interp_mc(
  y,
  compact.to,
  drop.zeros = FALSE,
  to.data.frame = FALSE,
  round.off = NULL,
  weight = NULL,
  mc.cores = 1
)
```

Arguments

y	A numeric vector, matrix or data frame.
compact.to	Proportion of remaining points after compression. If equals to 1 and keep.zeros = TRUE, the original vector is presented.
drop.zeros	Logical. Drop repeated zeros? Default: FALSE.
to.data.frame	Logical. Convert to data frame? Default: FALSE.
round.off	Number of decimal places of the interpolated y. Default: NULL.
weight	Vector of weights with same length of y. Default: NULL.
mc.cores	The number of cores to mclapply. Default: 1.

Value

A list of x and y convoluted values with length near to compact.to*length(y).

See Also

rm0, interp, interp_df

Examples

```
library(voice)
# Same result of interp() function if x is a vector
interp(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = FALSE)
interp_mc(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = FALSE)

interp(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
```

```
interp_mc(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = TRUE)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)


# getting Media data frame
M <- voice::extract_features(dirname(path2wav), mc.cores = 1, verbose = FALSE)

M.num <- M[,-(1:3)]
nrow(M.num)
cm1 <- interp_mc(M.num, compact.to = 0.1, drop.zeros = TRUE,
to.data.frame = FALSE, mc.cores = 1)
names(cm1)
lapply(cm1$f0, length)
```

---

is.audio *Match string for audio suffix*

Description

Given a character vector, returns a logical vector indicating which elements have a valid audio file extension.

Usage

```
is.audio(x)
```

Arguments

x	A character vector.

---

is.hosted *Return strings with a URL scheme*

Description

Given a character vector, returns a logical indicating whether the URLs in the vector respond without error.

Usage

```
is.hosted(x)
```

Arguments

x	A character vector.

is.local *Return strings without a URL scheme*

Description

Given a character vector, returns a logical indicating whether the paths in the vector point to existing local files.

Usage

```
is.local(x)
```

Arguments

x	A character vector.

is.url *Match string for URL prefix*

Description

Given a character vector, returns a logical vector indicating which elements have a URL scheme.

Usage

```
is.url(x)
```

Arguments

x	A character vector.

is.video *Match string for video suffix*

Description

Given a character vector, returns a logical vector indicating which elements have a valid video file extension.

Usage

```
is.video(x)
```

Arguments

x	A character vector.

is_mono *Verify if an audio is mono*

Description

Verify if an audio is mono

Usage

```
is_mono(x)
```

Arguments

x	Path to WAV audio file.

Value

Logical. 'TRUE' indicates a mono (one-channel) file. 'FALSE' indicates a non-mono (two-channel) file.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

is_mono(path2wav[1])
sapply(path2wav, is_mono)
```

media-files *These functions are sourced from the 'embedr' package by Michael McCarthy, under MIT License: https://github.com/mccarthy-m-g/embedr/blob/master/LICENSE.md This inclusion is temporary and will be discontinued once 'embedr' is available on CRAN. See https://github.com/mccarthy-m-g/embedr for more details.*

Description

Example Media Files

Usage

```
mp3
```

Format

An object of class `character` of length 1.

Details

Example media files included with embedr.

- ‘mp3‘: MP3 audio - ‘mp4‘: MP4 video - ‘png‘: PNG thumbnail

`mozilla_id_path` *Sample IDs and paths*

Description

A dataset containing sample IDs and paths from Ardila et al (2019) ’Common voice: A massively-multilingual speech corpus’, used in Zabala (2023) ’voice: new approaches to audio analysis’. The considered sample contains 34,425 rows associated with 838 IDs (p_s = 2.4%).

Usage

```
mozilla_id_path
```

References

Ardila R, Branson M, Davis K, Henretty M, Kohler M, Meyer J, Morais R, Saunders L, Tyers FM, Weber G (2019). "Common voice: A massively-multilingual speech corpus." arXiv preprint arXiv:1912.06670. URL https://arxiv.org/abs/1912.06670.

See Also

`extract_features`.

Examples

```
library(voice)
mozilla_id_path
```

notes *Assign notes to frequencies*

Description

Returns a vector of notes for equal-tempered scale, A4 = 440 Hz.

Usage

```
notes(x, method = "spn", moving.average = FALSE, k = 11)
```

Arguments

x	Numeric vector of frequencies in Hz.
method	Method of specifying musical pitch. (Default: spn, i.e., Scientific Pitch Notation).
moving.average	Logical. Must apply moving average? (Default: FALSE).
k	Integer width of the rolling window used if moving.average is TRUE. (Default: 11).

Details

The symbol '#' is being used to represent a sharp note, the higher in pitch by one semitone on Scientific Pitch Notation (SPN).

Value

A vector containing the notes for equal-tempered scale, A4 = 440 Hz. When 'method = 'spn'' the vector is of class 'ordered factor'. When 'method = 'octave'' the vector is of class 'factor'. When 'method = 'midi'' the vector is of class 'integer'.

References

https://gist.github.com/nvictor/7b4ab7070e210bc1306356f037226dd9

See Also

notes_freq

Examples

```
library(voice)
notes(c(220,440,880))
notes(c(220,440,880), method = 'octave')
notes(c(220,440,880), method = 'midi')
```

notes_freq *Frequencies on Scientific Pitch Notation (SPN)*

Description

Returns a tibble of frequencies on Scientific Pitch Notation (SPN) for equal-tempered scale, A4 = 440 Hz.

Usage

```
notes_freq()
```

Details

The symbol '#' is being used to represent a sharp note, the higher in pitch by one semitone. The SPN is also known as American Standard Pitch Notation (ASPN) or International Pitch Notation (IPN).

Value

A tibble with frequencies for equal-tempered scale, A4 = 440 Hz.

References

https://gist.github.com/nvictor/7b4ab7070e210bc1306356f037226dd9

See Also

notes

Examples

```
library(voice)
notes_freq()
```

piano_plot *Piano plot*

Description

Piano plot showing the notes in Scientific Pitch Notation.

Usage

```
piano_plot(data, num_fmt = 0)
```

Arguments

data	Data frame or tibble containing the desired frequencies to be plotted.
num_fmt	Number of the desired formant (includes f0 for simplicity). Default: num_fmt = 0.

References

https://en.wikipedia.org/wiki/12_equal_temperament

https://en.wikipedia.org/wiki/Scientific_pitch_notation

Examples

```
library(voice)
# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)
# Media dataset
M <- extract_features(path2wav)
piano_plot(M, 0)
piano_plot(M, 0:2)
```

---

read_rttm *Read RTTM files*

Description

Read Rich Transcription Time Marked (RTTM) files in fromRttm directory.

Usage

```
read_rttm(fromRttm)
```

Arguments

fromRttm	A directory/folder containing RTTM files.

Details

The Rich Transcription Time Marked (RTTM) files are space-delimited text files containing one turn per line defined by NIST - National Institute of Standards and Technology. Each line containing ten fields:

type Type: segment type; should always by SPEAKER.

file File ID: file name; basename of the recording minus extension (e.g., rec1_a).

chnl Channel ID: channel (1-indexed) that turn is on; should always be 1.

tbeg Turn Onset – onset of turn in seconds from beginning of recording.

tdur Turn Duration – duration of turn in seconds.

ortho Orthography Field – should always by <NA>.

stype Speaker Type – should always be <NA>.

name Speaker Name – name of speaker of turn; should be unique within scope of each file.

conf Confidence Score – system confidence (probability) that information is correct; should always be <NA>.

slat Signal Lookahead Time – should always be <NA>.

Value

A list containing data frames obtained from standard RTTM files. See 'Details'.

References

https://www.nist.gov/system/files/documents/itl/iad/mig/KWS15-evalplan-v05.pdf

See Also

voice::enrich_rttm

Examples

```
library(voice)

url0 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock0.rttm'
download.file(url0, destfile = paste0(tempdir(), '/sherlock0.rttm'))
url1 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock1.rttm'
download.file(url0, destfile = paste0(tempdir(), '/sherlock1.rttm'))

(rttm <- voice::read_rttm(tempdir()))
class(rttm)
lapply(rttm, class)
```

---

rm0 *Compress zeros.*

Description

Transforms n sets of m>n zeros (alternated with sets of non zeros) into n sets of n zeros.

Usage

```
rm0(y)
```

Arguments

y	A vector or time series.

Value

Vector with n zeros.

Examples

```
library(voice)

(v0 <- c(1:20,rep(0,10)))
(r0 <- rm0(v0))
length(v0)
length(r0)
sum(v0 == 0)

(v1 <- c(rep(0,10),1:20))
(r1 <- rm0(v1))
length(r1)

(v2 <- rep(0,10))
(r2 <- rm0(v2))
length(r2)

(v3 <- c(0:10))
(r3 <- rm0(v3))
length(r3)

(v4 <- c(rep(0,10), 1:10, rep(0,5), 10:20, rep(0,10)))
(r4 <- rm0(v4))
length(r4)
sum(v4 == 0)
```

---

smooth_df *Smooth numeric variables in a data frame*

Description

Smooth numeric variables in a data frame

Usage

```
smooth_df(x, k = 11, id = colnames(x)[1], colnum = NULL, mc.cores = 1)
```

Arguments

x	A data frame.
k	Integer width of the rolling window. Default: 11.
id	The identification column. Default: colname of the first column of x.
colnum	A char vector indicating the numeric colnames. If NULL, uses the columns of the numeric class.
mc.cores	The number of cores to mclapply. By default uses 1.

Value

Vector of interpolated values with length near to `compact.to*length(x)`.

See Also

`extract_features`

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

# minimal usage
M <- extract_features(path2wav, features = c('f0', 'fmt'))
(Ms <- smooth_df(M[-(1:2)]))
dim(M)
dim(Ms)
```

`splitw` *Split Wave*

Description

Split WAV files either in `fromWav` directory or using (same names) RTTM files/subdirectories as guidance.

Usage

```
splitw(
  fromWav,
  fromRttm = NULL,
  toSplit = NULL,
  autoDir = FALSE,
  subDir = FALSE,
  output = "wave",
  filesRange = NULL,
  full.names = TRUE,
  recursive = FALSE,
  silence.gap = 0.5
)
```

Arguments

fromWav	Either WAV file or directory containing WAV files.
fromRttm	Either RTTM file or directory containing RTTM files. Default: NULL.
toSplit	A directory to write generated files. Default: NULL.
autoDir	Logical. Must the directories tree be created? Default: FALSE. See 'Details'.
subDir	Logical. Must the splitted files be placed in subdirectories? Default: FALSE.
output	Character string, the class of the object to return, either 'wave' or 'list'.
filesRange	The desired range of directory files (default: NULL, i.e., all files). Must be TRUE only if fromWav is a directory.
full.names	Logical. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned. (default: TRUE) Used by base::list.files.
recursive	Logical. Should the listing recursively into directories? (default: FALSE) Used by base::list.files. Inactive if fromWav is a file.
silence.gap	The silence gap (in seconds) between adjacent words in a keyword. Rows with tdur <= silence.gap are removed. (default: 0.5)

Details

When autoDir = TRUE, the following directories are created: '../mp3','../rttm', '../split' and '../musicxml'. Use getwd() to find the parent directory '../'.

Value

Splited audio files according to the correspondent RTTM file(s). See 'voice::diarize'.

See Also

voice::diarize

Examples

```
## Not run:
library(voice)

urlWav <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/wav/sherlock0.wav'
destWav <- paste0(tempdir(), '/sherlock0.wav')
download.file(urlWav, destfile = destWav)

urlRttm <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock0.rttm'
destRttm <- paste0(tempdir(), '/sherlock0.rttm')
download.file(urlRttm, destfile = destRttm)

splitDir <- paste0(tempdir(), '/split')
dir.create(splitDir)
splitw(destWav, fromRttm = destRttm, toSplit = splitDir)
```

```
dir(splitDir)

## End(Not run)
```

---

spn2abc *Convert SPN to ABC*

Description

Convert SPN to standard octave.

Usage

```
spn2abc(x, to_lower = FALSE, spacing = TRUE)
```

Arguments

x	A vector containing a note in SPN (Scientific Pitch Notation).
to_lower	Logical. Should the string be lower case? Default: FALSE.
spacing	Logical. Should the strin return spaces between notes? Default: TRUE.

References

https://en.wikipedia.org/wiki/Scientific_pitch_notation

https://en.wikipedia.org/wiki/ABC_notation

Examples

```
library(voice)
spn2abc('C4')
spn2abc('C5')
spn2abc('C4', to_lower = TRUE)
spn2abc(c('C4','D#7','E2'))
spn2abc(c('C4','D#7','E2'), to_lower = TRUE)
spn2abc(c('C4','D#7','E2'), spacing = FALSE)
spn2abc(c('C4','D#7','E2'), to_lower = TRUE, spacing = FALSE)
```

tag *Tag a data frame with media information*

Description

Tag a data frame with media information

Usage

```
tag(
  x,
  groupBy = "wav_path",
  wavPath = unique(x$wav_path),
  wavPathName = "wav_path",
  tags = c("feat_summary"),
  sortByGroupBy = TRUE,
  filesRange = NULL,
  features = "f0",
  sex = "u",
  windowShift = 5,
  numFormants = 8,
  numcep = 12,
  dcttype = c("t2", "t1", "t3", "t4"),
  fbtype = c("mel", "htkmel", "fcmel", "bark"),
  resolution = 40,
  usecmp = FALSE,
  mc.cores = 1,
  full.names = TRUE,
  recursive = FALSE,
  check.mono = FALSE,
  stereo2mono = FALSE,
  overwrite = FALSE,
  freq = 44100,
  round.to = 4,
  verbose = FALSE
)
```

Arguments

x	An Extended data frame to be tagged with media information. See references.
groupBy	A variable to group the summary measures. The argument must be a character vector. (Default: groupBy = 'wav_path').
wavPath	A vector containing the path(s) to WAV files. May be both as dirname or basename formats.
wavPathName	A string containing the WAV path name. (Default: wavPathName = 'wav_path').
tags	Tags to be added to x. See Details. (Default: 'feat_summary').

sortByGroupBy	Logical. Should the function sort the Extended data frame x by gropuBy? (Default: sortByGroupBy = TRUE).
filesRange	The desired range of directory files. Should only be used when all the WAV files are in the same folder. (Default: NULL, i.e., all files).
features	Vector of features to be extracted. (Default: 'f0').
sex	= <code> set sex specific parameters where <code> = 'f'[emale], 'm'[ale] or 'u'[nknown] (default: 'u'). Used as 'gender' by wrassp::ksvF0, wrassp::forest and wrassp::mhsF0.
windowShift	= <dur> set analysis window shift to <dur>ation in ms (default: 5.0). Used by wrassp::ksvF0, wrassp::forest, wrassp::mhsF0, wrassp::zcrana, wrassp::rfcana, wrassp::acfana, wrassp::cepstrum, wrassp::dftSpectrum, wrassp::cssSpectrum and wrassp::lpsSpectrum.
numFormants	= <num> <num>ber of formants (Default: 8). Used by wrassp::forest.
numcep	Number of Mel-frequency cepstral coefficients (cepstra) to return (Default: 12). Used by tuneR::melfcc.
dcttype	Type of DCT used. 't1' or 't2', 't3' for HTK 't4' for feacalc (Default: 't2'). Used by tuneR::melfcc.
fbtype	Auditory frequency scale to use: 'mel', 'bark', 'htkmel', 'fcmel' (Default: 'mel'). Used by tuneR::melfcc.
resolution	= <freq> set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (Default: 40.0). Used by wrassp::cssSpectrum, wrassp::dftSpectrum and wrassp::lpsSpectrum.
usecmp	Logical. Apply equal-loudness weighting and cube-root compression (PLP instead of LPC) (Default: FALSE). Used by tuneR::melfcc.
mc.cores	Number of cores to be used in parallel processing. (Default: 1)
full.names	Logical. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned. (Default: TRUE) Used by base::list.files.
recursive	Logical. Should the listing recursively into directories? (Default: FALSE) Used by base::list.files.
check.mono	Logical. Check if the WAV file is mono. (Default: TRUE)
stereo2mono	(Experimental) Logical. Should files be converted from stereo to mono? (Default: TRUE)
overwrite	(Experimental) Logical. Should converted files be overwritten? If not, the file gets the suffix _mono. (Default: FALSE)
freq	Frequency in Hz to write the converted files when stereo2mono=TRUE. (Default: 44100)
round.to	Number of decimal places to round to. (Default: NULL)
verbose	Logical. Should the running status be showed? (Default: FALSE)

Details

filesRange should only be used when all the WAV files are in the same folder.

Value

A tibble data frame containing summarized numeric columns using (1) mean, (2) standard deviation, (3) variation coefficient, (4) median, (5) interquartile range and (6) median absolute deviation.

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern = glob2rx('*.wav'), full.names = TRUE)

# creating Extended synthetic data
E <- dplyr::tibble(subject_id = c(1,1,1,2,2,2,3,3,3),
wav_path = path2wav)
E

# minimal usage
tag(E)

# canonical data
tag(E, groupBy = 'subject_id')

# limiting filesRange
tag(E, filesRange = 3:6)

# more features
Et <- tag(E, features = c('f0', 'fmt', 'rf', 'rcf', 'rpf', 'rfc', 'mfcc'),
groupBy = 'subject_id')
Et
str(Et)
```

---

url.exists

These functions are sourced from the 'embedr' package by Michael McCarthy, under MIT License: https://github.com/mccarthy-m-g/embedr/blob/master/LICENSE.md This inclusion is temporary and will be discontinued once 'embedr' is available on CRAN. See https://github.com/mccarthy-m-g/embedr for more details.

Description

Check if URL exists

Usage

```
url.exists(x)
```

Arguments

x	A character vector.

Details

Given a character string, returns a logical vector indicating whether a request for a specific URL responds without error.

Value

'TRUE' if the URL responds without error, otherwise 'FALSE'.

`write_list` *Writes a list to a path*

Description

Writes a list to a path

Usage

```
write_list(x, path)
```

Arguments

`x`	A list.
`path`	A full path to file.

Value

A file named 'list.txt' in 'path'.

Examples

```
## Not run:
library(voice)

pts <- list(x = cars[,1], y = cars[,2])
listFile <- paste0(tempdir(), '/list.txt')
voice::write_list(pts, listFile)
file.info(listFile)
system(paste0('head ', listFile))

## End(Not run)
```

Index