

Package ‘voice’

September 2, 2022

Type Package

Title Tools for Voice Analysis, Speaker Recognition and Mood Inference

Version 0.4.14

Date 2022-09-01

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

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

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

Description General purpose tools for voice analysis, speaker recognition and mood inference. Gathers 'R' and 'Python' tools to solve problems concerning voice and audio in general.

Depends R (>= 4.1.0)

Imports dplyr, R.utils, reticulate, seewave, tibble, tidyselect, tuneR, wrassp, zoo

Suggests knitr

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Zabala Filipe J. [cre, aut]

Repository CRAN

Date/Publication 2022-09-02 07:50:02 UTC

R topics documented:

audio_time	2
conv	3
conv_df	4
conv_mc	6
duration	7
enrich_rttm	8
expand_model	9
extract_features	10
extract_features_py	12
feat_summary	13
get_bit	16
get_dur	17
get_left	17
get_right	18
get_samp.rate	19
get_tbeg	19
get_tdur	20
id_path	21
is_mono	21
notes	22
notes_freq	23
poetry	24
read_rttm	25
rm0	26
smooth_df	27
splitw	28
tag	29
write_list	32
Index	33

audio_time	<i>Returns the total time of audio files in seconds</i>
------------	---

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 <code>base::list.files</code> .

Value

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

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)
```

conv

Convolute vectors.

Description

Convolute vectors.

Usage

```
conv(
  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 <code>keep.zeros = T</code> , 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 convoluted vector. Default: NULL.
 weight Vector of weights with same length of y. Default: NULL.

Value

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

See Also

rm0, conv_mc, conv_df

Examples

```
library(voice)

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

# with weight
(c2 <- conv(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)
conv(v2, 0.1, drop.zeros = TRUE, to.data.frame = FALSE)
conv(v2, 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
conv(v2, 0.2, drop.zeros = TRUE)
conv(v2, 0.2, drop.zeros = FALSE)

(v3 <- c(rep(0,10), 1:20, rep(0,3)))
(c3 <- conv(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 <- conv(v4, 1/4, round.off = 3))
```

conv_df

Convolute data frames using multicore.

Description

Convolute data frames using multicore.

Usage

```
conv_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	Percentage of remaining points after compaction If equals to 1 and keep.zeros = T, 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 compress to 1 zero per null set? Default: FALSE.
to.data.frame	Logical. Should the return be a data frame? If FALSE returns a list. Default: TRUE.
round.off	Number of decimal places of the convoluted vector. Default: NULL.
weight	Vector of weights with same length of y. Default: NULL.
mc.cores	The number of cores to mclapply. By default uses 1.

Value

A vector of convoluted values with length near to compact.to*length(x).

See Also

conv, conv_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
```

```
M <- extract_features(dirname(path2wav), features = c('f0','formants'),
mc.cores = 1, verbose = FALSE)

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

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

conv_mc

Convolute vectors using multicore.

Description

Convolute vectors using multicore.

Usage

```
conv_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	Percentage of remaining points after compression. If equals to 1 and keep.zeros = T, 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 convoluted vector. 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, conv, conv_df

Examples

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

conv(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
conv_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 all features
M <- extract_features(dirname(path2wav), features = c('f0', 'formants',
'zcr', 'mhs', 'rms', 'gain', 'rfc', 'ac', 'cep', 'dft', 'css', 'lps', 'mfcc'),
mc.cores = 1, verbose = FALSE)

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

duration

Gives the duration of sequences.

Description

Gives the 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','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)
```

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

<code>listRttm</code>	A list containing RTTM files.
<code>silence.gap</code>	The silence gap (in seconds) between adjacent words in a keyword. Rows with <code>tdur <= silence.gap</code> are removed. (default: 0.5)
<code>as.tibble</code>	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'
download.file(url0, destfile = paste0(tempdir(), '/sherlock0.rttm'))
url1 <- 'https://raw.githubusercontent.com/filipezabala/voiceAudios/main/rttm/sherlock1.rttm'
download.file(url1, destfile = paste0(tempdir(), '/sherlock1.rttm'))

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

expand_model	<i>Expand model given y and x variables.</i>
--------------	--

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 *Extracts features from WAV audio files*

Description

Extracts features from WAV audio files.

Usage

```
extract_features(
  x,
  filesRange = NULL,
  features = c("f0", "formants", "mfcc", "df", "pf", "rf", "rcf", "rpf"),
  gender = "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 = NULL,
  verbose = TRUE
)
```

Arguments

x	A vector containing either files or directories of audio files in WAV format.
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.
features	Vector of features to be extracted. (default: 'f0','formants','mfcc','df','pf','rf','rcf','rpf'). The following features may contain a variable number of columns: 'cep', 'dft', 'css' and 'lps'.
gender	= <code> set gender specific parameters where <code> = 'f'[emale], 'm'[ale] or 'u'[nknown] (default: 'u'). Used 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	Logical. Should files be converted from stereo to mono? (default: TRUE)
overwrite	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

When features 'df', 'pf', 'rf', 'rcf' or 'rpf' are selected, 'f0' and 'formants' must be selected. 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 (2022), 'rcf' to 'formant cumulated removal' (rcf2:rcf8) by Zabala (2022) and 'rpf' to 'formant position removal' (rpf1:rpf8) by Zabala (2022).

Value

A Media data frame containing the selected features.

References

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)

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))
```

extract_features_py	<i>Extract features from WAV audios using 'Python's' 'Parselmouth' library.</i>
---------------------	---

Description

Extract features from WAV audios using 'Python's' 'Parselmouth' library.

Usage

```
extract_features_py(
  directory,
  filesRange = 0,
  features = c("f0", "formants"),
  windowShift = 5/1000,
  full.names = TRUE,
  recursive = FALSE,
  round.to = NULL
)
```

Arguments

directory	A directory/folder containing WAV files.
filesRange	The desired range of directory files (default: 0, i.e., all files).
features	Vector of features to be extracted. (default: 'f0' (pitch), 'formants' (F1:F8)).
windowShift	= <dur> set analysis window shift to <dur>ation in ms (default: 5/1000).
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)
recursive	Logical. Should the listing recursively into directories? (default: FALSE)
round.to	Number of decimal places to round to. (default: NULL)

Details

The function uses the getwd() folder to write the temp files.

Value

A data frame containing the selected features.

Examples

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

path2wav <- list.files(system.file('extdata', package = 'wrassp'),
  pattern <- glob2rx('*.*wav'), full.names = TRUE)
efp <- extract_features_py(dirname(path2wav))
efp
table(efp$file_name)

# limiting filesRange
efpl <- extract_features_py(dirname(path2wav), filesRange = 3:6)
efpl
table(efpl$file_name)

## End(Not run)
```

 feat_summary

Returns summary measures of 'voice::extract_features'

Description

Returns summary measures of 'voice::extract_features'

Usage

```

feat_summary(
  x,
  groupBy = "wav_path",
  wavPath = unique(x$wav_path),
  wavPathName = "wav_path",
  filesRange = NULL,
  features = "f0",
  gender = "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 = TRUE
)

```

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'
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.
features	Vector of features to be extracted. (default: 'f0', 'formants', 'mfcc', 'df', 'pf', 'rf', 'ref', 'rpf'). The following features may contain a variable number of columns: 'cep', 'dft', 'css' and 'lps'.
gender	= <code> set gender specific parameters where <code> = 'f'[emale], 'm'[ale] or 'u'[nknown] (default: 'u'). Used 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	Logical. Should files be converted from stereo to mono? (default: TRUE)
overwrite	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: TRUE)

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 mean, standard deviation, variation coefficient, media, interquartile range and 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, '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

An integer scalar 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)

rw1 <- lapply(path2wav, tuneR::readWave)
sapply(rw1, voice::get_bit)
```

get_dur	<i>Time duration</i>
---------	----------------------

Description

Get time duration from WAV file.

Usage

```
get_dur(x)
```

Arguments

x Wave object from 'tuneR::readWave'.

Value

A numeric scalar 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	<i>Get left channel</i>
----------	-------------------------

Description

Get left channel from WAV file.

Usage

```
get_left(x)
```

Arguments

x Wave object from 'tuneR::readWave'.

Value

A 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

A 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	<i>Get sample rate</i>
---------------	------------------------

Description

Get sample rate from WAV file.

Usage

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

Arguments

x Wave object from 'tuneR::readWave'.

Value

An integer scalar 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	<i>Time beginning</i>
----------	-----------------------

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

A 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

A 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)
```

id_path	<i>Sample IDs and paths</i>
---------	-----------------------------

Description

A dataset containing sample IDs and paths used in Zabala (2022) voice: new approaches to audio analysis. The considered sample contains 34,425 rows associated with 838 IDs (p_s = 2.4%).

Usage

```
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](https://arxiv.org/abs/1912.06670). URL <https://arxiv.org/abs/1912.06670>.

See Also

[extract_features](#).

Examples

```
library(voice)
id_path
```

is_mono	<i>Verify if an audio is mono.</i>
---------	------------------------------------

Description

Verify if an audio is mono.

Usage

```
is_mono(x)
```

Arguments

x Path to WAV audio file.

Value

A logical value. '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)
```

notes

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

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	A 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?
k	Integer width of the rolling window used if moving.average is TRUE.

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://pages.mtu.edu/~suits/notefreqs.html>

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

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://pages.mtu.edu/~suits/notefreqs.html>

Examples

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

 poetry

Poetry. The best words in their best order.

Description

Diarization from WAV audios using 'Python's' 'pyannote-audio' library.

Usage

```
poetry(
  fromWav,
  toRttm = NULL,
  autoDir = FALSE,
  pycall = "~/miniconda3/envs/pyvoice38/bin/python3.8"
)
```

Arguments

fromWav	A directory/folder containing WAV files.
toRttm	A directory/folder to write RTTM files. If the default toRttm = NULL is used, './voiceAudios/rttm' is created and used.
autoDir	Logical. Must the directories tree must be created? Default: FALSE. See 'Details'.
pycall	Python call.

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::poetry(fromWav = unique(dirname(wavDir)), toRttm = tempdir())
dir(tempdir())

## End(Not run)
```

read_rttm	<i>Read RTTM files</i>
-----------	------------------------

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 be 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 be <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(url1, destfile = paste0(tempdir(), '/sherlock1.rttm'))

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

rm0	<i>Transforms n sets of m>n zeros (alternated with sets of non zeros) into n sets of n zeros.</i>
-----	--

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

A vector of convoluted values with length near to `compact.to*length(x)`.

See Also

`extract_features`

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/folder containing WAV files.
fromRttm	Either RTTM file or directory/folder containing RTTM files. Default: NULL.
toSplit	A directory/folder 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

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

Examples

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

wavDir <- list.files(system.file('extdata', package = 'wrassp'),
                    pattern <- glob2rx('*.*wav'), full.names = TRUE)
splitDir <- paste0(tempdir(), '/split')
voice::poetry(fromWav = unique(dirname(wavDir)), toRttm = tempdir())
dir.create(splitDir)
dir(tempdir())

splitw(unique(dirname(wavDir)), fromRttm = tempdir(), toSplit = paste0(tempdir(), '/split'))
dir(splitDir)

## End(Not run)
```

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",
  gender = "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 = TRUE
)

```

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: <code>groupBy = 'wav_path'</code> .
wavPath	A vector containing the path(s) to WAV files. May be both as <code>dirname</code> or <code>basename</code> formats.
wavPathName	A string containing the WAV path name. Default: <code>wavPathName = 'wav_path'</code>
tags	Tags to be added to x. Default: <code>'feat_summary'</code> . See details.
sortByGroupBy	Logical. Should the function sort the Extended data frame x by <code>groupBy</code> ? Default: <code>sortByGroupBy = TRUE</code> .
filesRange	The desired range of directory files (default: <code>NULL</code> , i.e., all files). Should only be used when all the WAV files are in the same folder.
features	Vector of features to be extracted. (default: <code>'f0','formants','mfcc','df','pf','rf','rcf','rpf'</code>). The following features may contain a variable number of columns: <code>'cep'</code> , <code>'dft'</code> , <code>'css'</code> and <code>'lps'</code> .
gender	= <code><code></code> set gender specific parameters where <code><code> = 'f'</code> [emale], <code>'m'</code> [ale] or <code>'u'</code> [nknown] (default: <code>'u'</code>). Used by <code>wrassp::ksvF0</code> , <code>wrassp::forest</code> and <code>wrassp::mhsF0</code> .
windowShift	= <code><dur></code> set analysis window shift to <code><dur></code> ation in ms (default: 5.0). Used by <code>wrassp::ksvF0</code> , <code>wrassp::forest</code> , <code>wrassp::mhsF0</code> , <code>wrassp::zcrana</code> , <code>wrassp::rfcana</code> , <code>wrassp::acfana</code> , <code>wrassp::cepstrum</code> , <code>wrassp::dftSpectrum</code> , <code>wrassp::cssSpectrum</code> and <code>wrassp::lpsSpectrum</code> .
numFormants	= <code><num></code> <code><num></code> ber of formants (default: 8). Used by <code>wrassp::forest</code> .
numcep	Number of Mel-frequency cepstral coefficients (cepstra) to return (default: 12). Used by <code>tuneR::melfcc</code> .
dcttype	Type of DCT used. <code>'t1'</code> or <code>'t2'</code> , <code>'t3'</code> for HTK <code>'t4'</code> for <code>feacalc</code> (default = <code>'t2'</code>). Used by <code>tuneR::melfcc</code> .
fbtype	Auditory frequency scale to use: <code>'mel'</code> , <code>'bark'</code> , <code>'htkmel'</code> , <code>'fcmel'</code> (default: <code>'mel'</code>). Used by <code>tuneR::melfcc</code> .
resolution	= <code><freq></code> set FFT length to the smallest value which results in a frequency resolution of <code><freq></code> Hz or better (default: 40.0). Used by <code>wrassp::cssSpectrum</code> , <code>wrassp::dftSpectrum</code> and <code>wrassp::lpsSpectrum</code> .
usecmp	Logical. Apply equal-loudness weighting and cube-root compression (PLP instead of LPC) (default: <code>FALSE</code>). Used by <code>tuneR::melfcc</code> .

<code>mc.cores</code>	Number of cores to be used in parallel processing. (default: 1)
<code>full.names</code>	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 <code>base::list.files</code> .
<code>recursive</code>	Logical. Should the listing recursively into directories? (default: FALSE) Used by <code>base::list.files</code> .
<code>check.mono</code>	Logical. Check if the WAV file is mono. (default: TRUE)
<code>stereo2mono</code>	Logical. Should files be converted from stereo to mono? (default: TRUE)
<code>overwrite</code>	Logical. Should converted files be overwritten? If not, the file gets the suffix <code>_mono</code> . (default: FALSE)
<code>freq</code>	Frequency in Hz to write the converted files when <code>stereo2mono=TRUE</code> . (default: 44100)
<code>round.to</code>	Number of decimal places to round to. (default: NULL)
<code>verbose</code>	Logical. Should the running status be showed? (default: TRUE)

Details

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

Value

A tibble data frame containing summarized columns by the mean, standard deviation, variation coefficient, median, interquartile range and 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', 'formants', 'df', 'pf', 'rf', 'rcf', 'rpf'),
  groupBy = 'subject_id')
```

```
Et  
str(Et)
```

write_list	<i>Writes a list to a path.</i>
------------	---------------------------------

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])  
voice::write_list(pts, paste0(getwd(), '/list.txt'))  
  
## End(Not run)
```


Index

[audio_time](#), [2](#)

[conv](#), [3](#)

[conv_df](#), [4](#)

[conv_mc](#), [6](#)

[duration](#), [7](#)

[enrich_rttm](#), [8](#)

[expand_model](#), [9](#)

[extract_features](#), [10](#), [27](#)

[extract_features_py](#), [12](#)

[feat_summary](#), [13](#)

[get_bit](#), [16](#)

[get_dur](#), [17](#)

[get_left](#), [17](#)

[get_right](#), [18](#)

[get_samp.rate](#), [19](#)

[get_tbeg](#), [19](#)

[get_tdur](#), [20](#)

[id_path](#), [21](#)

[is_mono](#), [21](#)

[notes](#), [22](#)

[notes_freq](#), [23](#)

[poetry](#), [24](#)

[read_rttm](#), [25](#)

[rm0](#), [26](#)

[smooth_df](#), [27](#)

[splitw](#), [28](#)

[tag](#), [29](#)

[write_list](#), [32](#)