Package 'wrassp'

May 8, 2026

Version 1.0.6

Date 2025-12-17

Title Interface to the 'ASSP' Library

Description
A wrapper around Michel Scheffers's 'libassp' (<https://libassp.sourceforge.net/>). The 'libassp' (Advanced Speech Signal Processor) library aims at providing functionality for handling speech signal files in most common audio formats and for performing analyses common in phonetic science/speech science. This includes the calculation of formants, fundamental frequency, root mean square, auto correlation, a variety of spectral analyses, zero crossing rate, filtering etc. This wrapper provides R with a large subset of 'libassp's signal processing functions and provides them to the user in a (hopefully) user-friendly manner.

Encoding UTF-8

Depends R (>= 3.1.1), tibble(>= 2.1.0)

VignetteBuilder knitr

Suggests compare(>= 0.2.4), rmarkdown, knitr, testthat(>= 0.7.1)

License GPL (>= 3)

URL https://github.com/IPS-LMU/wrassp

BugReports https://github.com/IPS-LMU/wrassp/issues

RoxygenNote 7.3.3

NeedsCompilation yes

Author Markus Jochim [cre, aut] (ORCID:
<https://orcid.org/0000-0002-5638-4870>),
Raphael Winkelmann [aut],
Lasse Bombien [aut],
Michel Scheffers [aut]

Maintainer Markus Jochim <markusjochim@phonetik.uni-muenchen.de>

Repository CRAN

Date/Publication 2025-12-18 06:10:10 UTC

Contents

`wrassp-package` *wrassp - Interface to the ASSP Library*

Description

wrassp is a wrapper for R around Michel Scheffers's libassp (Advanced Speech Signal Processor). The libassp library aims at providing functionality for handling speech signal files in most common audio formats and for performing analyses common in phonetic science/speech science. This includes the calculation of formants, fundamental frequency, root mean square, auto correlation, a variety of spectral analyses, zero crossing rate, filtering etc. This wrapper provides R with a large subset of libassp's signal processing functions and provides them to the user in a (hopefully) user-friendly manner.

Details

This package is part of the next iteration of the EMU Speech Database Management System which aims to be as close to an all-in-one solution for generating, manipulating, querying, analyzing and managing speech databases as possible. For an overview of the system please visit this URL: http://ips-lmu.github.io/EMU.html.

Available signal processing functions:

1. acfana: Analysis of short-term autocorrelation function
2. afdiff: Computes the first difference of the signal
3. affilter: Filters the audio signal (see docs for types)
4. cepstrum: Short-term cepstral analysis
5. cssSpectrum: Cepstral smoothed version of dftSpectrum
6. dftSpectrum: Short-term DFT spectral analysis
7. forest: Formant estimation
8. ksvF0: F0 analysis of the signal
9. lpsSpectrum: Linear Predictive smoothed version of dftSpectrum
10. mhsF0: Pitch analysis of the speech signal using Michel's (M)odified (H)armonic (S)ieve algorithm
11. rfcana: Linear Prediction analysis
12. rmsana: Analysis of short-term Root Mean Square amplitude
13. zcrana: Analysis of the averages of the short-term positive and negative zero-crossing rates

Available file handling functions:

1. read.AsspDataObj: read an existing SSFF file into a AsspDataObj which is its in-memory equivalent.
2. write.AsspDataObj: write a AsspDataObj out to a SSFF file.

Author(s)

Maintainer: Markus Jochim <markusjochim@phonetik.uni-muenchen.de> (ORCID)

Authors:

- Raphael Winkelmann <raphael@phonetik.uni-muenchen.de>
- Lasse Bombien <lasse@phonetik.uni-muenchen.de>
- Michel Scheffers

See Also

Useful links:

- https://github.com/IPS-LMU/wrassp
- Report bugs at https://github.com/IPS-LMU/wrassp/issues

`acfana` *acfana*

Description

acfana function adapted from libassp

Usage

```
acfana(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  windowShift = 5,
  windowSize = 20,
  effectiveLength = TRUE,
  window = "BLACKMAN",
  analysisOrder = 0,
  energyNormalization = FALSE,
  lengthNormalization = FALSE,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

`listOfFiles`	vector of file paths to be processed by function
`optLogFilePath`	path to option log file
`beginTime`	= <time>: set begin of analysis interval to <time> seconds (default: 0 = beginning of file)
`centerTime`	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules BeginTime, EndTime and WindowShift options
`endTime`	= <time>: set end of analysis interval to <time> seconds (default: 0 = end of file)

`windowShift` = <dur>: set analysis window shift to <dur> ms (default: 5.0)

`windowSize` = <dur>: set analysis window size to <dur> ms; overrules EffectiveLength parameter

`effectiveLength` make window size effective rather than exact

`window` = <type>: set analysis window function to <type> (default: BLACKMAN)

`analysisOrder` = <num>: set analysis order to <num> (default: 0 = sample rate in kHz + 3)

`energyNormalization` calculate energy-normalized autocorrelation

`lengthNormalization` calculate length-normalized autocorrelation

`toFile` write results to file (default extension is .acf)

`explicitExt` set if you wish to override the default extension

`outputDirectory` directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files

`forceToLog` is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.

`verbose` display infos & show progress bar

Details

Analysis of short-term autocorrelation function of the signals in <listOFFiles>. Analysis results will be written to a file with the base name of the input file and extension '.acf'. Default output is in SSFF binary format (track 'acf').

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate short-term autocorrelation
res <- acfana(path2wav, toFile=FALSE)

# plot short-term autocorrelation values
matplot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
```

```
attr(res, 'startTime'),
res$acf,
type='l',
xlab='time (s)',
ylab='short-term autocorrelation values')
```

---

addTrack *Add a track to an AsspDataObj*

Description

Add a track to an AsspDataObj

Usage

```
addTrack(dobj, trackname, data, format = "INT16", deleteExisting = FALSE)
```

Arguments

dobj	The data object to which the data is to be added
trackname	The name of the new track
data	a matrix with values
format	format for binary writing to file (defaults to 'INT16')
deleteExisting	Delete existing track with the same (default: FALSE)

Details

The specified data object is extended by a new track named `trackname`. If there already is a track with the same name and `deleteExisiting` is `FALSE` the function does nothing but returns with an error. If `deleteExisting` is `TRUE` the existing track will be removed (see `delTrack`. `data` to be added is a numeric matrix (or will be coerced to one). It must have the same number of rows as the tracks that already exist in the object (if any). TODO add `format` information.

Value

the object including the new track

Author(s)

Lasse Bombien

See Also

`delTrack`

afdiff *afdiff*

Description

afdiff function adapted from libassp

Usage

```
afdiff(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  computeBackwardDifference = FALSE,
  computeCentralDifference = FALSE,
  channel = 1,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
computeBackwardDifference	compute backward difference (s'[n] = s[n] - s[n-1]) (default: forward difference s'[n] = s[n+1] - s[n])
computeCentralDifference	compute central/interpolated/3-point difference
channel	= <num>: for multi-channel input files: extract and differentiate channel <num> (1 <= <num> <= 8 default: channel 1)
toFile	write results to file (default extension is .d+(extensionsOfAudioFile))
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Computes the first difference of the signal in the audio- formatted file(s) <listOfFiles>. The differentiated signal will be written to a file with the base name of the input file and an extension consisting of '.d', followed by the extension of the input file. The format of the output file will be the same as that of the input file. Differentiation can improve results on F0 analysis of e.g. EGG signals because it removes a DC offset, attenuates very low frequency components - and in the case of central differentiation also very high ones - and enhances the moment of glottal closure.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# compute the first forward difference of the signal
res <- afdiff(path2wav, toFile=FALSE)

# plot samples
# (only plot every 10th element to accelerate plotting)
plot(seq(0,numRecs.AsspDataObj(res) - 1, 10) / rate.AsspDataObj(res),
     res$audio[c(TRUE, rep(FALSE,9))],
     type='l',
     xlab='time (s)',
     ylab='Audio samples')
```

`affilter` *affilter*

Description

affilter function adapted from libassp

Usage

```
affilter(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  highPass = 4000,
  lowPass = 0,
  stopBand = 96,
  transition = 250,
  useIIR = FALSE,
  numIIRsections = 4,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
highPass	= <num>: set the high-pass cut-off frequency to <num> Hz (default: 4000, high-pass filtering is applied)
lowPass	= <num>: set the low-pass cut-off frequency to <num> Hz (default: 0, no low-pass filtering)
stopBand	= <num>: set the stop-band attenuation to <num> dB (default: 93.0 dB, minimum: 21.0 dB)
transition	= <num>: set the width of the transition band to <num> Hz (default: 250.0 Hz)
useIIR	switch from the default FIR to IIR filter
numIIRsections	= <num>: set the number of 2nd order sections to <num> (default: 4) where each section adds 12dB/oct to the slope of the filter
toFile	write results to file (for default extension see details section))
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Filters the audio signal in <listOfFiles>. By specifying the high-pass and/or low-pass cut-off frequency one of four filter characteristics may be selected as shown in the table below.

hp	lp	filter characteristic	extension
> 0	0	high-pass from hp	'.hpf'
0	> 0	low-pass up to lp	'.lpf'
> 0	> hp	band-pass from hp to lp	'.bpf'
> lp	> 0	band-stop between lp and hp	'.bsf'

Please note: per default a high-pass filter from 0 to 4000 Hz is applied.

The Kaiser-window design method is used to compute the coefficients of a linear-phase FIR filter with unity gain in the pass-band. The cut-off frequencies (-6 dB points) of the filters are in the middle of the transition band. The filtered signal will be written to a file with the base name of the input file and an extension corresponding to the filter characteristic (see table). The format of the output file will be the same as that of the input file.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# band-pass filter signal between 4000 and 5000 Hz
res <- affilter(path2wav, highPass=4000, lowPass=5000, toFile=FALSE)

# plot samples
# (only plot every 10th element to accelerate plotting)
plot(seq(0,numRecs.AsspDataObj(res) - 1, 10) / rate.AsspDataObj(res),
     res$audio[c(TRUE, rep(FALSE,9))],
     type='l',
     xlab='time (s)',
     ylab='Audio samples')
```

`AsspDataFormat` *Get/set data format of an AsspDataObj*

Description

Function to get or set the data format of an AsspDataObj.

Usage

```
AsspDataFormat(x)

AsspDataFormat(x)  <- value
```

Arguments

x	an object of class AsspDataObj
value	an integer or a string indicating the new data format

Details

libassp can store data in binary and ASCII format. This function enables the user to determine the data format of an object read from file and to set it for subsequent writing. Valid values are 'ascii' (or 1) for ASCII format or 'binary' (or 2) for binary IO. Use is discouraged unless the user knows what they are doing.

Value

a string representing the current data format

for AsspDataFormat<-, the updated object

Author(s)

Lasse Bombien

See Also

AsspFileFormat

AsspFileFormat *Get and set AsspFileFormat*

Description

Function to get or set the file format of an AsspDataObj.

Usage

```
AsspFileFormat(x)

AsspFileFormat(x)  <- value
```

Arguments

x	an object of class AsspDataObj
value	an integer or a string indicating the new file format

Details

libassp handles a number of file formats common in speech research. This function enables the user to determine the file format of an object read from file and to set it for subsequent writing. This allows for file format conversion to some degree. Note, that many conversions are not reasonable/possible: conversions are therefore discouraged unless the user knows what they are doing. Format specifiers can be found in AsspFileFormats and exist in two forms: a code name and a code number. Both are suitable for setting the format.

Value

for AsspFileFormat the code name of the object's currently set file format

for AsspFileFormat<-, the updated object

Author(s)

Lasse Bombien

See Also

AsspFileFormats, AsspDataFormat

Examples

```
## Not run:
obj  <- read.AsspDataObj('/path/to/file.wav')
AsspFileFormat(obj)
AsspFileFormat(obj) <- 'SSFF' ## or
AsspFileFormat(obj) <- 20

## End(Not run)
```

AsspFileFormats *list of possibly useful file formats for AsspDataObj corresponding to the first element of the fileInfo attribute*

Description

list of possibly useful file formats for AsspDataObj corresponding to the first element of the fileInfo attribute

Usage

```
AsspFileFormats
```

Format

Code Name	code number	description
RAW	1	headerless or unsupported format
ASP_A	2	ASP with ASCII data
ASP_B	3	ASP with binary data
XASSP	4	xassp ASCII
IPDS_M	5	labels in IPdS 'MIX' format
IPDS_S	6	labels in IPdS 'SAMPA' format
AIFF	7	Apple Audio Interchange File Format
AIFC	8	AIFF extended for compressed data
CSL	9	Kay Elemetrics Computerized Speech Lab
CSRE	10	Computerized Speech Research Environment
ESPS	11	Entropic Signal Processing System
ILS	12	
KTH	13	Kungliga Tekniska Hoegskolan Stockholm
SWELL	13	commercial version of KTH
SNACK	13	as Tcl extension
SFS	14	University College London Speech Filing System
SND	15	NeXT version of 'SND' format
AU	15	Sun version of 'SND' format
NIST	16	National Institute of Standards and Technology
SPHERE	16	SPeech HEader REsources
PRAAT_S	17	UvA praat 'short' text file
PRAAT_L	18	UvA praat 'long' text file
PRAAT_B	19	UvA praat binary file
SSFF	20	Simple Signal File Format
WAVE	21	IBM/Microsoft RIFF-WAVE
WAVE_X	22	RIFF-WAVE extended format (Revision 3)
XLABEL	24	ESPS xlabel
YORK	25	University of York (Klatt'80 parameters)
UWM	26	University of Wisconsin at Madison (microbeam data))

Author(s)

Lasse Bombien

See Also

AsspFileFormat

AsspLpTypes *AsspLpTypes*

Description

returns all valid AsspLpTypes according to the assp library

Usage

```
AsspLpTypes()
```

Details

wrapper function for AsspLpTypes of wrassp

Value

vector containing lp types

Author(s)

Raphael Winkelmann

AsspSpectTypes *AsspSpectTypes*

Description

returns all valid AsspSpectTypes according to the assp library

Usage

```
AsspSpectTypes()
```

Details

wrapper function for AsspSpectTypes of wrassp

Value

vector containing spectrogram types

Author(s)

Raphael Winkelmann

`AsspWindowTypes` *AsspWindowTypes*

Description

returns all valid AsspWindowTypes according to the assp library

Usage

```
AsspWindowTypes()
```

Details

wrapper function for AsspWindowTypes of wrassp

Value

vector containing window types

Author(s)

Raphael Winkelmann

`cepstrum` *cepstrum*

Description

calculate cepstral coefficients using libassp

Usage

```
cepstrum(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  resolution = 40,
  fftLength = 0,
  windowShift = 5,
  window = "BLACKMAN",
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default: begin of data)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of data)
resolution	= <freq>: set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (default: 40.0)
fftLength	= <num>: set FFT length to <num> points (overrules default and 'resolution' option)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
toFile	write results to file (default extension depends on)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Short-term cepstral analysis of the signal in <listOfFiles> using the Fast Fourier Transform. The number of coefficients per output record will also equal the FFT length / 2 + 1 (i.e. be non-mirrored). Analysis results will be written to a file with the base name of the input file and as extension '.cep'. Default output is in SSFF format with 'cep' as track name.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

See Also

dftSpectrum, cssSpectrum, lpsSpectrum; all derived from libassp's spectrum function

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calulate cepstrum
res <- cepstrum(path2wav, toFile=FALSE)

# plot cepstral values at midpoint of signal
plot(res$cep[dim(res$cep)[1]/2,],
     type='l',
     xlab='cepstral value index',
     ylab='cepstral value')
```

---

cssSpectrum *cssSpectrum*

Description

calculate cepstrally smoothed spectrum using libassp

Usage

```
cssSpectrum(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  resolution = 40,
  fftLength = 0,
  windowShift = 5,
  window = "BLACKMAN",
  numCeps = 0,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file

beginTime	= <time>: set begin of analysis interval to <time> seconds (default: begin of data)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of data)
resolution	= <freq>: set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (default: 40.0)
fftLength	= <num>: set FFT length to <num> points (overrules default and 'resolution' option)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
numCeps	= <num>: set number of cepstral coefficients used to <num> (default: sampling rate in kHz + 1; minimum: 2)
toFile	write results to file (default extension depends on)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Short-term spectral analysis of the signal in <listOfFiles> using the Fast Fourier Transform and cepstral smoothing. Analysis results will be written to a file with the base name of the input file and '.css.' as extension. Default output is in SSFF format with 'css' in lower case as track name.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

See Also

dftSpectrum, lpsSpectrum, cepstrum; all derived from libassp's spectrum function.

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate cepstrally smoothed spectrum
res <- cssSpectrum(path2wav, toFile=FALSE)

# plot spectral values at midpoint of signal
plot(res$css[dim(res$css)[1]/2,],
     type='l',
     xlab='spectral value index',
     ylab='spectral value')
```

---

delTrack *Remove track from an AsspDataObj*

Description

Remove a track from an AsspDataObj object

Usage

```
delTrack(dobj, trackname)
```

Arguments

dobj	An object of class AsspDataObj
trackname	the name of a track in this object

Value

The object without the track named trackname

Author(s)

Lasse Bombien

dftSpectrum *dftSpectrum*

Description

DFT spectrum function adapted from libassp

Usage

```
dftSpectrum(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  resolution = 40,
  fftLength = 0,
  windowShift = 5,
  window = "BLACKMAN",
  bandwidth = 0,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default: begin of data)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of data)
resolution	= <freq>: set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (default: 40.0)
fftLength	= <num>: set FFT length to <num> points (overrules default and 'resolution' option)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
bandwidth	= <freq>: set the effective analysis bandwidth to <freq> Hz (default: 0, yielding the smallest possible value given the length of the FFT)

toFile	write results to file (default extension depends on)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Short-term spectral analysis of the signal in <listOfFiles> using the Fast Fourier Transform. The default is to calculate an unsmoothed narrow-band spectrum with the size of the analysis window equal to the length of the FFT. The output from the FFT will be converted to a power spectrum in dB from 0 Hz up to and including the Nyquist rate. Analysis results will be written to a file with the base name of the input file and the spectrum type in lower case as extension (e.g. '.dft'). Default output is in SSFF format with the spectrum type in lower case as track name.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

See Also

cssSpectrum, lpsSpectrum, cepstrum; all derived from libassp's spectrum function.

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate dft spectrum
res <- dftSpectrum(path2wav, toFile=FALSE)

# plot spectral values at midpoint of signal
plot(res$dft[dim(res$dft)[1]/2,],
     type='l',
     xlab='spectral value index',
     ylab='spectral value')
```

`dur.AsspDataObj` *Timing information on AsspDataObj*

Description

Various information on AsspDataObj

Usage

```
dur.AsspDataObj(x)

numRecs.AsspDataObj(x)

rate.AsspDataObj(x)

startTime.AsspDataObj(x)
```

Arguments

`x`	an object of class AsspDataObj

Details

Some utility function to retrieve duration, number of records, sample rate and so on.

Value

dur: the duration of the AsspDataObj in ms

numRecs: the number of records stored in the AsspDataObj

rate: the data/sample rate of the AsspDataObj in Hz

startTime: start time of the first sample of the AsspDataObj

Author(s)

Lasse Bombien

forest *forest*

Description

forest function adapted from libassp

Usage

```
forest(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  endTime = 0,
  windowShift = 5,
  windowSize = 20,
  effectiveLength = TRUE,
  nominalF1 = 500,
  gender = "m",
  estimate = FALSE,
  order = 0,
  incrOrder = 0,
  numFormants = 4,
  window = "BLACKMAN",
  preemphasis = -0.8,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default = 0: begin of data)
endTime	= <time>: set end of analysis interval to <time> seconds (default = 0: end of data)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
windowSize	= <dur>: set analysis window size to <dur> ms (default: 30.0)
effectiveLength	make window size effective rather than exact
nominalF1	= <freq>: set nominal F1 frequency to <freq> Hz (default: 500.0 Hz)

gender	= <code>: set gender specific parameters where <code> = f[emale], m[ale] or u[nknown] (when <code>=f: eff. window length = 12.5 ms nominal F1 = 560.0 Hz)
estimate	insert rough frequency estimates of missing formants (default: frequency set to zero)
order	decrease default order by 2 (one resonance less)
incrOrder	increase default order by 2 (one resonance more)
numFormants	= <num>: set number of formants to <num> (default: 4; maximum: 8 or half the LP order)
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
preemphasis	= <val>: set pre-emphasis factor to <val> (-1 <= val <= 0) (default: dependent on sample rate and nominal F1)
toFile	write results to file (default extension is .fms)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Formant estimation of the signal(s) in <listOfFiles>. Raw resonance frequency and bandwidth values are obtained by root-solving of the Linear Prediction polynomial from the autocorrelation method and the Split-Levinson-Algorithm (SLA). Resonances are then classified as formants using the so-called Pisarenko frequencies (by-product of the SLA) and a formant frequency range table derived from the nominal F1 frequency. The latter may have to be increased by about 12% for female voices (see NominalF1 and Gender options). Formant estimates will be written to a file with the base name of the input file and extension '.fms'. Default output is in SSFF binary format (tracks 'fm' and 'bw')

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

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

```
                        full.names = TRUE)[1]

# calculate formant values
res <- forest(path2wav, toFile=FALSE)

# plot formant values
matplot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
          attr(res, 'startTime'),
        res$fm,
        type='l',
        xlab='time (s)',
        ylab='Formant frequency (Hz)')
```

`is.AsspDataObj` *Checks whether x is a valid AsspDataObj*

Description

Checks whether x is a valid AsspDataObj

Usage

```
is.AsspDataObj(x, ...)
```

Arguments

`x`	an object of class AsspDataObj
`...`	optional other arguments passed to further functions

Value

TRUE or FALSE

Author(s)

Lasse Bombien

isAsspLpType *isAsspLpType*

Description

checks if given string is a valid AsspLpType according to the assp library

Usage

```
isAsspLpType(lpName = NULL)
```

Arguments

lpName	name of lp type

Value

(BOOL) true if lpName is valid; false otherwise

Author(s)

Raphael Winkelmann

isAsspSpectType *isAsspSpectType*

Description

checks if given string is a valid AsspSpectType according to the assp library

Usage

```
isAsspSpectType(spectName = NULL)
```

Arguments

spectName	name of lp type

Value

(BOOL) true if spectName is valid; false otherwise

Author(s)

Raphael Winkelmann

isAsspWindowType *isAsspWindowType*

Description

checks if given string is a valid AsspWindowType according to the assp library

Usage

```
isAsspWindowType(windowName = NULL)
```

Arguments

windowName	name of window

Value

(BOOL) true if windowName is valid; false otherwise

Author(s)

Raphael Winkelmann

ksvF0 *ksvF0*

Description

ksvF0 function adapted from libassp

Usage

```
ksvF0(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  endTime = 0,
  windowShift = 5,
  gender = "u",
  maxF = 600,
  minF = 50,
  minAmp = 50,
  maxZCR = 3000,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
```

```
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default = 0: begin of data)
endTime	set end of analysis interval to <time> seconds (default = 0: end of data)
windowShift	= <dur>: set frame shift to <dur> ms (default: 5.0)
gender	= <code> set gender-specific F0 ranges; <code> may be: "f[emale]" (80.0 - 640.0 Hz) "m[ale]" (50.0 - 400.0 Hz) "u[nknown]" (default; 50.0 - 600.0 Hz)
maxF	= <freq>: set maximum F0 value to <freq> Hz (default: 500.0)
minF	= <freq>: set minimum F0 value to <freq> Hz (default: 50.0)
minAmp	= <amp>: set amplitude threshold for voiced samples to <amp> (default: 100)
maxZCR	maximum zero crossing rate in Hz (for voicing detection)
toFile	write results to file (default extension is .f0)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

F0 analysis of the signal in <listOfFiles> using the K. Schaefer-Vincent periodicity detection algorithm. Analysis results will be written to a file with the base name of the input file and extension '.f0'. Default output is in SSFF binary format (track 'F0'). Optionally, location and type of the signal extrema on which the F0 data are based, may be stored in a label file. The name of this file will consist of the base name of the F0 file and the extension '.prd'.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

References

Schaefer-Vincent K (1983) Pitch period detection and chaining: method and evaluation. Phonetica 1983, Vol 40, pp. 177-202

See Also

mhsF0 for an alternative pitch tracker

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate fundamental frequency contour
res <- ksvF0(path2wav, toFile=FALSE)

# plot the fundamental frequency contour
plot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
       attr(res, 'startTime'),
     res$F0,
     type='l',
     xlab='time (s)',
     ylab='F0 frequency (Hz)')
```

---

lpsSpectrum *lpsSpectrum*

Description

Calculate Linear Prediction smoothed spectrum using libassp

Usage

```
lpsSpectrum(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  resolution = 40,
  fftLength = 0,
  windowSize = 20,
  windowShift = 5,
  window = "BLACKMAN",
  order = 0,
```

```
  preemphasis = -0.95,
  deemphasize = TRUE,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default: begin of data)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of data)
resolution	= <freq>: set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (default: 40.0)
fftLength	= <num>: set FFT length to <num> points (overrules default and 'resolution' option)
windowSize	= <dur>: set effective analysis window size to <dur> ms
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
order	= <num>: set prediction order to <num> (default: sampling rate in kHz + 3)
preemphasis	= <val>: set pre-emphasis factor to <val> (default: -0.95)
deemphasize	(default: undo spectral tilt due to pre-emphasis used in LP analysis, i.e. TRUE)
toFile	write results to file (default extension depends on)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Short-term spectral analysis of the signal in <listOfFiles> using the Fast Fourier Transform and linear predictive smoothing. Analysis results will be written to a file with the base name of the input file and the spectrum type in lower case as extension (i.e. '.lps'). Default output is in SSFF format with the spectrum type in lower case as track name.

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

See Also

dftSpectrum, cssSpectrum, cepstrum; all derived from libassp's spectrum function.

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate lps spectrum
res <- lpsSpectrum(path2wav, toFile=FALSE)

# plot spectral values at midpoint of signal
plot(res$lps[dim(res$lps)[1]/2,],
     type='l',
     xlab='spectral value index',
     ylab='spectral value')
```

---

mhsF0 *mhsF0*

Description

mhsF0 function adapted from libassp

Usage

```
mhsF0(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  windowShift = 5,
  gender = "u",
  maxF = 600,
  minF = 50,
```

```
  minAmp = 50,
  minAC1 = 0.25,
  minRMS = 18,
  maxZCR = 3000,
  minProb = 0.52,
  plainSpectrum = FALSE,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default = 0: begin of file)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default = 0: end of file)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
gender	= <code> set gender-specific pitch ranges; <code> may be: "f[emale]" (80.0 - 600.0 Hz) "m[ale]" (50.0 - 375.0 Hz) "u[nknown]" (default; 50.0 - 600.0 Hz)
maxF	= <freq>: set maximum pitch value to <freq> Hz (default: 500.0)
minF	= <freq>: set minimum pitch value to <freq> Hz (default: 50.0 minimum: 25.0)
minAmp	= <amp>: minimum signal amplitude (default: 50)
minAC1	= <freq>: minimum 1st correlation coefficient (default: 0.250)
minRMS	= <num>: minimum RMS amplitude in dB (default: 18.0)
maxZCR	= <freq>: maximum zero crossing rate in Hz (default: 3000)
minProb	= <num>: minimum quality value of F0 fit (default: 0.520)
plainSpectrum	use plain rather than masked power spectrum
toFile	write results to file (default extension is .pit)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Pitch analysis of the speech signal in <listOfFile> using Michel's/Modified Harmonic Sieve algorithm. Analysis results will be written to a file with the base name of the input file and extension '.pit'. Default output is in SSFF binary format (track 'pitch').

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

See Also

`ksvF0` for an tracking the fundamental frequency

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate fundamental frequency contour
res <- mhsF0(path2wav, toFile=FALSE)

# plot fundamental frequency contour
plot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
       attr(res, 'startTime'),
     res$pitch,
     type='l',
     xlab='time (s)',
     ylab='F0 frequency (Hz)')
```

---

prepareFiles *Normalise a list of filenames so that they can be passed to a signal processing function*

Description

Normalise a list of filenames so that they can be passed to a signal processing function

Usage

```
prepareFiles(listOfFiles)
```

Arguments

listOfFiles	The list of file names to process

Value

A normalised list of filenames

Author(s)

Raphael Winkelmann

Examples

```
#   listOfFiles <- prepareFiles(listOfFiles)
```

---

print.AsspDataObj *print a summary of an AsspDataObj*

Description

Prints an overview of ASSP Data Objects

Usage

```
## S3 method for class 'AsspDataObj'
print(x, ...)
```

Arguments

x	an object of class AsspDataObj
...	other arguments that might be passed on to other functions

Author(s)

Lasse Bombien

See Also

read.AsspDataObj

`read.AsspDataObj` *read.AsspDataObj from a signal/parameter file*

Description

read.AsspDataObj creates an object of class dobj from a signal or parameter file readable by the ASSP Library (WAVE, SSFF, AU, ...)

Usage

```
read.AsspDataObj(fname, begin = 0, end = 0, samples = FALSE)
```

Arguments

`fname`	filename of the signal or parameter file
`begin`	begin time (default is in seconds) of segment to retrieve
`end`	end time (default is in seconds) of segment to retrieve
`samples`	(BOOL) if set to false seconds values of begin/end are sample numbers

Value

list object containing file data

Author(s)

Lasse Bombien

`rfcana` *rfcana*

Description

rfcana function adapted from libassp

Usage

```
rfcana(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  windowShift = 5,
  windowSize = 20,
  effectiveLength = TRUE,
```

```
  window = "BLACKMAN",
  order = 0,
  preemphasis = -0.95,
  lpType = "RFC",
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default = 0: begin of file)
centerTime	set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default = 0: end of file)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
windowSize	= <dur>: set analysis window size to <dur> ms; overrules effectiveLength option
effectiveLength	make window size effective rather than exact
window	= <type>: set analysis window function to <type> (default: BLACKMAN)
order	= <num>: set prediction order to <num> (default: sample rate in kHz + 3)
preemphasis	= <val>: set pre-emphasis factor to <val> (default: -0.95)
lpType	= <type>: calculate <type> LP parameters; <type> may be: "ARF": area function "LAR": log area ratios "LPC": linear prediction filter coefficients "RFC": reflection coefficients (default)
toFile	write results to file (default extension dependent on LpType .arf/.lar/.lpc/.rfc)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Linear Prediction analysis of <listOfFiles> using the autocorrelation method and the Durbin recursion. This program calculates the RMS amplitudes of the input and residual signal in dB and, per

default, reflection coefficients (see '-t' option). Analysis results will be written to a file with the base name of the input file and the parameter type in lower case as extension (e.g. '.rfc'). Default output is in SSFF binary format (tracks 'rms', 'gain' and the LP type in lower case).

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# perform linear prediction analysis
res <- rfcana(path2wav, toFile=FALSE)

# plot reflection coefficients
matplot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
          attr(res, 'startTime'),
        res$rfc,
        type='l',
        xlab='time (s)',
        ylab='reflection coefficient values')
```

`rmsana` *rmsana*

Description

rmsana function adapted from libassp

Usage

```
rmsana(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  windowShift = 5,
  windowSize = 20,
```

```
  effectiveLength = TRUE,
  linear = FALSE,
  window = "HAMMING",
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default = 0: begin of file)
centerTime	= <time>: set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of file)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
windowSize	= <dur>: set analysis window size to <dur> ms; overrules effectiveLength option
effectiveLength	make window size effective rather than exact
linear	calculate linear RMS values (default: values in dB)
window	= <type>: set analysis window function to <type> (default: HAMMING)
toFile	write results to file (default extension is .rms)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Analysis of short-term Root Mean Square amplitude of the signal in <listOfFiles>. Per default, the RMS values are expressed in decibel (dB) so that they correspond to the short-term power of the signal. Analysis results will be written to a file with the base name of the input file and extension '.rms'. Default output is in SSFF binary format (track 'rms').

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate rms values
res <- rmsana(path2wav, toFile=FALSE)

# plot rms values
plot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
       attr(res, 'startTime'),
     res$rms,
     type='l',
     xlab='time (s)',
     ylab='RMS energy (dB)')
```

---

tracks.AsspDataObj *tracks.AsspDataObj*

Description

List the tracks of an AsspDataObj

Usage

```
tracks.AsspDataObj(x)
```

Arguments

x	an object of class AsspDataObj

Details

AsspDataObj contain tracks (at least one). This function lists the names of these tracks. This function is equivalent to calling `names(x)`.

Value

a character vector containing the names of the tracks

Author(s)

Lasse Bombien

useWrasspLogger *package variable to force the usage of the logger set to FALSE by default*

Description

package variable to force the usage of the logger set to FALSE by default

Usage

```
useWrasspLogger
```

Format

An object of class `logical` of length 1.

Author(s)

Raphael Winkelmann

wrassp.logger *wrassp.logger*

Description

Designated logger for the wrassp signal processing functions

Usage

```
wrassp.logger(fName, fOpts, optLogFilePath, listOfFiles)
```

Arguments

`fName`	the name of the function calling the logger
`fOpts`	are the function options given by the user acquired by match.call
`optLogFilePath`	path to option log file
`listOfFiles`	vector of file paths that the spf calling the logger processed

Details

Function logs the call to a signal processing function (spf) of wrassp. It is called by default if the forceToLog option of the spf is not set to false. I tries to format the output in an easily readable fashion.

Author(s)

Raphael Winkelmann

See Also

match.call

---

wrasspOutputInfos *list of default output extensions, track names and output type for each signal processing function in wrassp*

Description

list of default output extensions, track names and output type for each signal processing function in wrassp

Usage

```
wrasspOutputInfos
```

Format

An object of class `list` of length 13.

Author(s)

Raphael Winkelmann

---

write.AsspDataObj *write.AsspDataObj to file*

Description

Writes an object of class AsspDataObj to a file given the meta information contained in the object.

Usage

```
write.AsspDataObj(dobj, file = attr(dobj, "filePath"))
```

Arguments

dobj	an object of class AsspDataObj
file	file name as a character string, defaults to the `filePath` attribute of the AsspDataObj

Author(s)

Lasse Bombien

zcrana *zcrana*

Description

zcrana function adapted from libassp

Usage

```
zcrana(
  listOfFiles = NULL,
  optLogFilePath = NULL,
  beginTime = 0,
  centerTime = FALSE,
  endTime = 0,
  windowShift = 5,
  windowSize = 25,
  toFile = TRUE,
  explicitExt = NULL,
  outputDirectory = NULL,
  forceToLog = useWrasspLogger,
  verbose = TRUE
)
```

Arguments

listOfFiles	vector of file paths to be processed by function
optLogFilePath	path to option log file
beginTime	= <time>: set begin of analysis interval to <time> seconds (default: begin of file)
centerTime	= <time> set single-frame analysis with the analysis window centred at <time> seconds; overrules beginTime, endTime and windowShift options
endTime	= <time>: set end of analysis interval to <time> seconds (default: end of file)
windowShift	= <dur>: set analysis window shift to <dur> ms (default: 5.0)
windowSize	= <dur>: set analysis window size to <dur> ms (default: 25.0)
toFile	write results to file (default extension is .zcr)
explicitExt	set if you wish to override the default extension
outputDirectory	directory in which output files are stored. Defaults to NULL, i.e. the directory of the input files
forceToLog	is set by the global package variable useWrasspLogger. This is set to FALSE by default and should be set to TRUE is logging is desired.
verbose	display infos & show progress bar

Details

Analysis of the averages of the short-term positive and negative zero-crossing rates of the signal in <listOfFiles>. Analysis results will be written to a file with the base name of the input file and extension '.zcr'. Default output is in SSFF binary format (track 'zcr').

Value

nrOfProcessedFiles or if only one file to process return AsspDataObj of that file

Author(s)

Raphael Winkelmann

Lasse Bombien

Examples

```
# get path to audio file
path2wav <- list.files(system.file("extdata", package = "wrassp"),
                       pattern = glob2rx("*.wav"),
                       full.names = TRUE)[1]

# calculate zcr values
res <- zcrana(path2wav, toFile=FALSE)

# plot zcr values
plot(seq(0,numRecs.AsspDataObj(res) - 1) / rate.AsspDataObj(res) +
       attr(res, 'startTime'),
     res$zcr,
     type='l',
     xlab='time (s)',
     ylab='ZCR values')
```

Index