Package 'reapeR'

August 24, 2026

Type Package

Title Interface to the 'REAPER' Library

Version 0.2.0

Description Wrapper for David Talkin's 'Robust Epoch and Pitch EstimatoR' (REAPER) software for estimating pitch and glottal closure instants from recordings of speech. For more information about the procedure, see <https://github.com/google/REAPER/tree/master>.

Encoding UTF-8

Depends R (>= 3.1.0)

Imports dplyr (>= 1.1.4), emuR (>= 2.5.2), readr (>= 2.1.5), stats (>= 4.5.1), tuneR (>= 1.4.7), wrassp (>= 1.0.5), Rcpp (>= 1.0.0)

LinkingTo Rcpp

RoxygenNote 7.3.3

License Apache License (>= 2)

URL https://github.com/rpuggaardrode/reapeR

BugReports https://github.com/rpuggaardrode/reapeR/issues

NeedsCompilation yes

Author Rasmus Puggaard-Rode [aut, cre],
David Talkin [ctb],
Google Inc. [cph]

Maintainer Rasmus Puggaard-Rode <rasmus.puggaard-rode@ling-phil.ox.ac.uk>

Repository CRAN

Date/Publication 2026-08-24 10:10:02 UTC

Contents

`read_epochs_out` *Read in REAPER epochs output file as tibble*

Description

Reads in an ASCII epochs / pitchmarks file created by REAPER as a vector of epochs

Usage

```
read_epochs_out(epochsFile, audioFile = NULL)
```

Arguments

`epochsFile`	String giving the file name of an ASCII epochs / pitchmarks file created by REAPER.
`audioFile`	String giving the name of the original audio file. If a string is provided, the function will return a named list instead of a vector (practical if creating a large object with many of these files). Default is 'NULL'.

Value

A numeric vector of epochs, or a named list with this vector.

Examples

```
path <- file.path(system.file(package = 'reapeR'), 'extdata', 'epochs')
epochs <- read_epochs_out(path)
```

`read_pitch_out` *Read in REAPER pitch output file as tibble*

Description

Reads in an ASCII pitch file created by REAPER as a well-formatted tibble

Usage

```
read_pitch_out(pitchFile, audioFile = NULL)
```

Arguments

pitchFile	String giving the file name of an ASCII pitch file created by REAPER.
audioFile	String giving the name of the original audio file. If a string is provided, a separate column will be made in the tibble repeating this string (practical if creating a large data frame with many of these files). Default is 'NULL', in which case no such column is created.

Value

A data frame with four columns: * 'time', giving the frame time in seconds * 'voiced' Boolean specifying whether that frame is 'voiced' * 'f0' F0 in Hz * 'file' String giving the name of the analyzed file.

Examples

```
path <- file.path(system.file(package = 'reapeR'), 'extdata', 'pitch')
pitch <- read_pitch_out(path)
```

---

reaper *Estimate pitch and/or epochs with REAPER*

Description

Track pitch and/or estimate epochs using David Talkin's REAPER library.

Usage

```
reaper(
  file,
  start = 0,
  end = Inf,
  channel = 1,
  f0min = 40,
  f0max = 500,
  interval = 0.005,
  hilbert = FALSE,
  suppress_highpass_filter = FALSE,
  unvoiced_cost = 0.9,
  output = c("pitch", "epochs"),
  force_list_output = FALSE,
  verbose = FALSE
)
```

Arguments

file	String giving the name of a WAV file to analyze
start	Numeric value giving the analysis start time in seconds. When possible, the actual analysis will begin 50 ms prior to this time. Default is '0'.
end	Numeric value giving the analysis end time in seconds. When possible, the actual analysis will begin 50 ms after this time. Default is 'Inf'.
channel	Numeric value specifying the channel to be analyzed. Default is '1'.
f0min	Numeric value specifying pitch floor. Default is '40'.
f0max	Numeric value specifying pitch ceiling. Default is '500'.
interval	Numeric value giving the F0 output interval in seconds. Default is '0.005'.
hilbert	Boolean; should Hilbert transform be applied prior to analysis? Default is 'FALSE'.
suppress_highpass_filter	Boolean; should highpass filter be suppressed? Default is 'FALSE'.
unvoiced_cost	Numeric; cost for unvoiced segments. Default is '0.9', set higher value to estimate more F0 in noise.
output	String or vector of strings specifying which estimates to output. Default is to output 'pitch' and 'epochs'. Possible values are: * 'pitch' Returning a data frame with equidistant pitch estimates at intervals determined by 'interval'. * 'epochs' returning a named list with a numeric vector specifying the times of estimated glottal closure instants. * 'gci_cand' Returning a data frame with times that are candidate glottal closure instants, as well as their time, local pitch value if determined to be voiced, and normalized cross-correlation functions ('nccf'). * 'probs' Returning a data frame with pseudo-probabilities of voicing, voicing onset, and voicing offset at regular intervals, as well as bandpassed (low-frequency) root-mean-squared amplitude of the signal, which is used to derive the pseudo-probability of voicing. * 'resids' Returning a data frame with the original signal, linear prediction residuals, and linear prediction residuals normalized by local amplitude.
force_list_output	Boolean; if 'TRUE', the function returns a named nested list even if only one output is chosen. Default is 'FALSE'.
verbose	Boolean; should diagnostic messages be printed to the console? Default is 'FALSE'.

Value

Depending on the value of 'output', either a nested list of outputs, a data frame, or a numeric vector.

Examples

```
snd <- file.path(system.file('extdata', package = 'reapeR'), '1.wav')
vals <- reaper(snd)
```

reaper2emuDB *Import REAPER output to an EMU database as SSFF files*

Description

When pitch has been tracked over an EMU database, this function is used to import the measures to the database in the Simple Signal File Format.

Usage

```
reaper2emuDB(reaper_output, db_handle, fileExtension = "reaper")
```

Arguments

reaper_output	Data frame or list object created with the 'reaper_bulk()' function run over a loaded EMU database.
db_handle	Handle of a loaded EMU database
fileExtension	String giving the desired file extension for the new SSFF files. Default is 'reaper'.

Value

Nothing; run for side effects.

Examples

```
emuR::create_emuRdemoData(tempdir())
db <- emuR::load_emuDB(file.path(tempdir(), 'emuR_demoData', 'ae_emuDB'))
emuR::list_ssffTrackDefinitions(db)
out <- reaper_bulk(db)
reaper2emuDB(out, db)
emuR::list_ssffTrackDefinitions(db)
```

reaper2ssff *Convert REAPER output to SSFF format*

Description

Convert REAPER output for a single file to Simple Signal File Format data. Usually used by [reaper2emuDB], but may also be used as a stand-alone for e.g. plotting REAPER pitch data with 'praatpicture'.

Usage

```
reaper2ssff(reaper_output)
```

Arguments

reaper_output	Output from [reaper] containing pitch measures.

Value

An object of class ‘AsspDataObj‘.

Examples

```
snd <- file.path(system.file('extdata', package = 'reapeR'), '1.wav')
vals <- reaper(snd)
ssffObj <- reaper2ssff(vals)
```

---

reaper_bulk *Bulk estimate pitch and/or epochs with with REAPER*

Description

Wrapper function to call David Talkin's REAPER software on all WAV files in a directory

Usage

```
reaper_bulk(
  directory,
  output = c("pitch", "epochs"),
  f0min = 40,
  f0max = 500,
  hirst2pass = FALSE,
  praat_output = FALSE,
  praat_output_dir = NULL,
  hirst2pass_f0min = 60,
  hirst2pass_f0max = 700,
  ...
)
```

Arguments

directory	String giving the name of a directory where all WAV files should be processed. Alternatively the handle of a loaded EMU database.
output	String or vector of strings specifying which estimates to output. Default is to output ‘pitch‘ and ‘epochs‘. Possible values are: * ‘pitch‘ Returning a data frame with equidistant pitch estimates at intervals determined by ‘interval‘. * ‘epochs‘ returning a named list with a numeric vector specifying the times of estimated glottal closure instants. * ‘gci_cand‘ Returning a data frame with times that are candidate glottal closure instants, as well as their time, local pitch value if determined to be voiced, and normalized cross-correlation functions (‘nccf‘). * ‘probs‘ Returning a data frame with pseudo-probabilities of voicing,

voicing onset, and voicing offset at regular intervals, as well as bandpassed (low-frequency) root-mean-squared amplitude of the signal, which is used to derive the pseudo-probability of voicing.

`f0min` Numeric value specifying pitch floor. Default is '40'.

`f0max` Numeric value specifying pitch ceiling. Default is '500'.

`hirst2pass` Boolean; should pitch floor and ceiling be dynamically estimated using the two-pass procedure proposed by Hirst and de Looze? In this case, REAPER is first run with liberal floor and ceiling values of 60 Hz and 700 Hz respectively, and then rerun with optimal floor and ceiling values estimated from the first and third quantiles of the first pass. (Floor = 0.75 Q1, ceiling = 1.5 Q3). If 'directory' refers to an EMU database with multiple sessions, the two-pass procedure is run separately for each session.

`praat_output` Boolean; should REAPER output be stored as a Praat '.Pitch' file? Default is 'FALSE'.

`praat_output_dir` String giving the location of a directory where Praat '.Pitch' files should be stored. Default is 'NULL'.

`hirst2pass_f0min` Numeric giving the pitch floor in Hz for the first pass in a two-pass procedure. Default is '60'.

`hirst2pass_f0max` Numeric giving the pitch ceiling in Hz for the first pass in a two-pass procedure. Default is '700'.

`...` Further arguments passed on to 'reaper()'.

Value

Depending on the value of 'output', either a nested list of outputs, a data frame, or a numeric vector.

Examples

```
dir <- file.path(system.file('extdata', package = 'reapeR'))
vals <- reaper_bulk(dir)
```

`reaper_wrap` *Wrap REAPER C++ library*

Description

Low-level wrapper for calling REAPER's 'epoch_tracker' procedure

Usage

```
reaper_wrap(
  samples,
  sample_rate,
  f0min = 40,
  f0max = 500,
  suppress_highpass_filter = FALSE,
  hilbert = FALSE,
  interval = 0.005,
  unvoiced_cost = 0.9,
  unvoiced_pulse_interval = 0.01,
  verbose = FALSE
)
```

Arguments

samples	Numeric vector giving audio samples in 16-bit integer format.
sample_rate	Numeric giving the sampling rate of ‘samples‘.
f0min	Numeric value specifying pitch floor. Default is ‘40‘.
f0max	Numeric value specifying pitch ceiling. Default is ‘500‘.
suppress_highpass_filter	Boolean; should highpass filter be suppressed? Default is ‘FALSE‘.
hilbert	Boolean; should Hilbert transform be applied prior to analysis? Default is ‘FALSE‘.
interval	Numeric value giving the F0 output interval in seconds. Default is ‘0.005‘.
unvoiced_cost	Numeric; cost for unvoiced segments. Default is ‘0.9‘, set higher value to estimate more F0 in noise.
unvoiced_pulse_interval	Numeric; what should be the interval in seconds of epoch pulses outside of voiced intervals? Default is ‘0.01‘.
verbose	Boolean; should messages be printed to the console?

Value

A list object with five elements: ‘epochs‘ gives the location of (voiced and unvoiced) epochs; ‘voicing‘ (same length as epochs) gives information about whether epochs are voiced; ‘f0‘ gives estimated pitch; ‘correlation‘ gives correlation between pitch estimates; ‘f0_interval‘ returns the value of ‘interval‘.

Examples

```
file <- file.path(system.file('extdata', package = 'reapeR'), '1.wav')
snd <- tuneR::readWave(file)
results <- reaper_wrap(snd@left, snd@samp.rate)
```

write_praat_epochs *Write REAPER output to Praat PointProcess file*

Description

Convert the estimated epoch locations of REAPER to a '.PointProcess' file that can be read by Praat.

Usage

```
write_praat_epochs(reaper_out, directory, filename)
```

Arguments

reaper_out	Data frame with the output of REAPER, as loaded with [read_epochs_out] or created with [reaper].
directory	String giving the location to use for storing the resulting '.PointProcess' file.
filename	String giving the file name of the resulting file (note that the '.PointProcess' extension is added automatically).

Value

Used for side effects.

Examples

```
snd <- file.path(system.file('extdata', package = 'reapeR'), '1.wav')
epo <- reaper(snd, output = 'epochs')
out_path <- file.path(tempdir(), 'epo.PointProcess')
file.exists(out_path)
write_praat_epochs(epo, tempdir(), 'epo')
file.exists(out_path)
```

write_praat_pitch *Write REAPER output to Praat Pitch file*

Description

Convert the pitch estimates of REAPER to a '.Pitch' file that can be read by Praat.

Usage

```
write_praat_pitch(reaper_out, directory, filename)
```

Arguments

reaper_out	Data frame with the output of REAPER, as loaded with [read_pitch_out] or created with [reaper].
directory	String giving the location to use for storing the resulting '.Pitch' file.
filename	String giving the file name of the resulting file (note that the '.Pitch' extension is added automatically).

Value

Used for side effects.

Examples

```
snd <- file.path(system.file('extdata', package = 'reapeR'), '1.wav')
pit <- reaper(snd, output = 'pitch')
out_path <- file.path(tempdir(), 'pit.Pitch')
file.exists(out_path)
write_praat_pitch(pit, tempdir(), 'pit')
file.exists(out_path)
```

Index