Package 'readtextgrid'

May 9, 2026

Type Package

Title Read in a 'Praat' 'TextGrid' File

Version 0.2.0

Description 'Praat' <https://www.fon.hum.uva.nl/praat/> is a widely used tool for manipulating, annotating and analyzing speech and acoustic data. It stores annotation data in a format called a 'TextGrid'. This package provides a way to read these files into R.

License GPL-3

Encoding UTF-8

Depends R (>= 4.3.0)

Suggests testthat (>= 2.1.0)

RoxygenNote 7.3.3

Imports utils, stats, tibble, purrr, readr, stringr, dplyr, rlang, withr

URL https://github.com/tjmahr/readtextgrid, https://www.tjmahr.com/readtextgrid/

BugReports https://github.com/tjmahr/readtextgrid/issues

LinkingTo cpp11

Config/Needs/website rmarkdown

NeedsCompilation yes

Author Tristan Mahr [aut, cre] (ORCID: <https://orcid.org/0000-0002-8890-5116>), Dan Villarreal [ctb], Jonathan Washington [ctb], Josef Fruehwald [aut]

Maintainer Tristan Mahr <tristan.mahr@wisc.edu>

Repository CRAN

Date/Publication 2025-10-27 20:00:09 UTC

Contents

`example_textgrid` *Locate the path of an example textgrid file*

Description

Locate the path of an example textgrid file

Usage

```
example_textgrid(which = 1)
```

Arguments

`which`	index of the textgrid to load

Details

This function is a wrapper over `system.file()` to locate the paths to bundled textgrids. These files are used to test or demonstrate functionality of the package.

Two files are included:

1. `"Mary_John_bell.TextGrid"` - the default TextGrid created by Praat's Create TextGrid command. This file is saved as UTF-8 encoding.
2. `"utf_16_be.TextGrid"` - a TextGrid with some IPA characters entered using Praat's IPA character selector. This file is saved with UTF-16 encoding.
3. `"nested-intervals.TextGrid"` - A textgrid containing an `"utterance"` tier, a `"words"` tier, and a `"phones"` tier. This file is typical of forced alignment textgrids where utterances contain words which contain speech segments. In this case, alignment was made by hand so that word and phone boundaries do not correspond exactly.

Value

Path of `"Mary_John_bell.TextGrid"` bundled with the `readtextgrid` package.

`pivot_textgrid_tiers` *Pivot a textgrid into wide format, respecting nested tiers*

Description

Pivot a textgrid into wide format, respecting nested tiers

Usage

```
pivot_textgrid_tiers(data, tiers, join_cols = "file")
```

Arguments

`data`	a textgrid dataframe created with `read_textgrid()`
`tiers`	character vector of tiers to pivot into wide format. When `tiers` has more than 1 element, the tiers are treated as nested. For example, if `tiers` is `c("utterance", "word", "phone")`, where `"utterance"` intervals contain `"word"` intervals which in turn contain `"phone"` intervals, the output will have one row per `"phone"` interval and include `utterance_*` and `word_*` columns for the utterance and word intervals that contain each phone interval. `tiers` should be ordered from broadest to narrowest (e.g, `"word"` preceding `"phone"`).
`join_cols`	character vector of the columns that will uniquely identify a textgrid file. Defaults to `"file"` because these columns have identical values for tiers read from the same textgrid file.

Details

For the joining nested intervals, two intervals *a* and *b* are combined into the same row if they match on the values in the `join_cols` columns and if the `a$xmin` <= `b$xmid` and `b$xmid` <= `a$xmax`. That is, if the midpoint of *b* is contained inside the interval *a*.

Value

a dataframe with just the intervals from tiers named in `tiers` converted into a wide format. Columns are renamed so that the `text` column is pivot into columns named after the tier names. For example, the `text` column in a `words` tier is renamed to `words`. The `xmax`, `xmin`, `annotation_num`, `tier_num`, `tier_type` are also prefixed with the tier name. For example, the `xmax` column in a `words` tier is renamed to `words_xmax`. An additional helper column `xmid` is added and prefixed appropriately. See examples below.

Examples

```
data <- example_textgrid(3) |>
  read_textgrid()
data

# With a single tier, we get just that tier with the columns prefixed with
```

```
# the tier_name
pivot_textgrid_tiers(data, "utterance")
pivot_textgrid_tiers(data, "words")

# With multiple tiers, intervals in one tier that contain intervals in
# another tier are combined into the same row.
a <- pivot_textgrid_tiers(data, c("utterance", "words"))
cols <- c(
  "utterance", "utterance_xmin", "utterance_xmax",
  "words", "words_xmin", "words_xmax"
)
a[cols]

a <- pivot_textgrid_tiers(data, c("utterance", "words", "phones"))
cols <- c(cols, "phones", "phones_xmin", "phones_xmax")
a[cols]
```

`read_textgrid` *Read a textgrid file into a tibble*

Description

Read a textgrid file into a tibble

Usage

```
read_textgrid(path, file = NULL, encoding = NULL)

read_textgrid_lines(lines, file = NULL)

legacy_read_textgrid(path, file = NULL, encoding = NULL)

legacy_read_textgrid_lines(lines, file = NULL)
```

Arguments

`path`	a path to a textgrid
`file`	an optional value to use for the `file` column. For `read_textgrid()`, the default is the base filename of the input file. For `read_textgrid_lines()`, the default is `NA`.
`encoding`	the encoding of the textgrid. The default value `NULL` uses `readr::guess_encoding()` to guess the encoding of the textgrid. If an encoding is provided, it is forwarded to `[readr::locale()]` and `[readr::read_lines()]`.
`lines`	alternatively, the lines of a textgrid file

Details

The `legacy_read_textgrid` functions are the original textgrid parsers provided by the package. They assume that the TextGrid file is a "long" format textgrid; this is the default format used by "Save a text file..." in Praat.

The current `read_textgrid()` functions are more flexible and can read in "short" format textgrids and textgrids with comments.

See https://www.fon.hum.uva.nl/praat/manual/TextGrid_file_formats.html for a description of the textgrid file format. Note that this package does not strictly adhere to format as described in this document. For example, the document says that numbers should be freestanding (surrounded by spaces or string boundaries), but Praat.exe can handle malformed numbers like `100ms`. Therefore, we tried to implement a parser that matched what Praat actually handles.

Value

a tibble with one row per textgrid annotation

Examples

```
tg <- system.file("Mary_John_bell.TextGrid", package = "readtextgrid")
read_textgrid(tg)
```

Index