

# Package ‘phonfieldwork’

November 23, 2020

**Type** Package

**Title** Linguistic Phonetic Fieldwork Tools

**Version** 0.0.10

**Depends** R (>= 3.5.0)

**Imports** tuneR, phonTools, grDevices, utils, graphics, rmarkdown, xml2,  
uchardet, tools

**Description** There are a lot of different typical tasks that have to be solved during phonetic research and experiments. This includes creating a presentation that will contain all stimuli, renaming and concatenating multiple sound files recorded during a session, automatic annotation in 'Praat' TextGrids (this is one of the sound annotation standards provided by 'Praat' software, see Boersma & Weenink 2020 <<https://www.fon.hum.uva.nl/praat/>>), creating an html table with annotations and spectrograms, and converting multiple formats ('Praat' TextGrid, 'ELAN', 'EXMARaLDA', 'Audacity', subtitles '.srt', and 'FLEx' flex-text). All of these tasks can be solved by a mixture of different tools (any programming language has programs for automatic renaming, and Praat contains scripts for concatenating and renaming files, etc.). 'phonfieldwork' provides a functionality that will make it easier to solve those tasks independently of any additional tools. You can also compare the functionality with other packages: 'rPraat' <<https://CRAN.R-project.org/package=rPraat>>, 'textgRid' <<https://CRAN.R-project.org/package=textgRid>>.

**License** GPL (>= 2)

**SystemRequirements** pandoc (>= 1.14) - <http://pandoc.org>

**URL** <https://CRAN.R-project.org/package=phonfieldwork>,  
<https://ropensci.github.io/phonfieldwork/>

**BugReports** <https://github.com/ropensci/phonfieldwork/issues>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**Suggests** knitr, dplyr, tidyr, DT, lingtypology, testthat

**Language** en-US

**NeedsCompilation** no

**Author** George Moroz [aut, cre] (<<https://orcid.org/0000-0003-1990-6083>>),  
Jonathan Keane [rev] (<<https://orcid.org/0000-0001-7087-9776>>),  
Niko Partanen [rev] (<<https://orcid.org/0000-0001-8584-3880>>)

**Maintainer** George Moroz <[agricolamz@gmail.com](mailto:agricolamz@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-11-23 14:10:02 UTC

## R topics documented:

add_leading_symbols . . . . .	3
annotate_textgrid . . . . .	3
audacity_to_df . . . . .	4
concatenate_soundfiles . . . . .	5
concatenate_textgrids . . . . .	6
create_empty_textgrid . . . . .	7
create_glossed_document . . . . .	8
create_image_look_up . . . . .	9
create_presentation . . . . .	9
create_sound_play . . . . .	11
create_subannotation . . . . .	11
create_viewer . . . . .	13
df_to_tier . . . . .	14
draw_sound . . . . .	15
draw_spectrogram . . . . .	18
eaf_to_df . . . . .	20
exb_to_df . . . . .	21
extract_intervals . . . . .	21
flextext_to_df . . . . .	22
formant_to_df . . . . .	23
get_sound_duration . . . . .	24
get_textgrid_names . . . . .	24
intensity_to_df . . . . .	25
pitch_to_df . . . . .	25
read_from_folder . . . . .	26
rename_soundfiles . . . . .	27
set_textgrid_names . . . . .	28
srt_to_df . . . . .	29
textgrid_to_df . . . . .	29
tier_to_df . . . . .	30

**Index**

**31**

---

add\_leading\_symbols     *Create indices padded with zeros*

---

**Description**

Create indices padded with zeros. This is important for creating appropriate for sorting names.

**Usage**

```
add_leading_symbols(file_names)
```

**Arguments**

file\_names     vector of any values.

**Value**

A string with numbers padded with leading zero.

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

annotate\_textgrid     *Annotate textgrid*

---

**Description**

Annotates textgrids. It is possible to define step in the argument "each", so each second element of the tier will be annotated.

**Usage**

```
annotate_textgrid(  
  annotation,  
  textgrid,  
  tier = 1,  
  each = 1,  
  backup = TRUE,  
  write = TRUE  
)
```

**Arguments**

annotation	vector of stimuli
textgrid	character with a filename or path to the TextGrid
tier	value that could be either ordinal number of the tier either name of the tier
each	non-negative integer. Each element of x is repeated each times
backup	logical. If TRUE (by default) it creates a backup tier.
write	logical. If TRUE (by default) it overwrites an existing tier.

**Value**

a string that contain TextGrid. If argument write is TRUE, then no output.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```

annotate_textgrid(
  annotation = c("", "t", "e", "s", "t"),
  textgrid = system.file("extdata",
    "test.TextGrid",
    package = "phonfieldwork"
  ),
  tier = 2, write = FALSE
)

```

---

audacity\_to\_df      *Audacity's labels to dataframe*

---

**Description**

Audacity make it possible to annotate sound files with labels that can be exported as a .tsv file with .txt extension. This function convert result to dataframe.

**Usage**

```
audacity_to_df(file_name)
```

**Arguments**

file\_name      file\_name string with a filename or path to the .txt file produced by Audacity

**Value**

a dataframe with columns: content, time\_start, time\_end, source.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
audacity_to_df(system.file("extdata",  
    "test_audacity.txt",  
    package = "phonfieldwork"  
))
```

---

concatenate\_soundfiles

*Concatenate sounds*

---

**Description**

Creates a merged sound file from old sound files in a folder. If the annotation argument is not equal to NULL, it creates an annotation file (Praat .TextGrid, ELAN .eaf or EXMARaLDA .exb) with original sound names annotation.

**Usage**

```
concatenate_soundfiles(  
  path,  
  result_file_name = "concatenated",  
  annotation = "textgrid"  
)
```

**Arguments**

path	path to the directory with soundfiles.
result_file_name	name of the result and annotation files.
annotation	character. There are several variants: "textgrid" for Praat TextGrid, "eaf" for ELAN's .eaf file, or "exb" for EXMARaLDA's .exb file. It is also possible to use NULL in order to prevent the creation of the annotation file.

**Value**

no output

**Author(s)**

George Moroz <agricolamz@gmail.com>

### Examples

```
# create two files in a temporary folder "test_folder"
s1 <- system.file("extdata", "test.wav", package = "phonfieldwork")
s2 <- system.file("extdata", "post.wav", package = "phonfieldwork")
tdir <- tempdir()
file.copy(c(s1, s2), tdir)

# here are two .wav files in a folder
list.files(tdir)
# [1] "post.wav" "test.wav" ...

# Concatenate all files from the folder into concatenated.wav and create
# corresponding TextGrid
concatenate_soundfiles(path = tdir, result_file_name = "concatenated")

list.files(tdir)
# [1] "concatenated.TextGrid" "concatenated.wav" "post.wav" "test.wav" ...
```

---

concatenate\_textgrids *Concatenate sounds*

---

### Description

Creates a merged sound file from old sound files in a folder. If the annotation argument is not equal to NULL, it creates an annotation file (Praat .TextGrid, ELAN .eaf or EXMARaLDA .exb) with original sound names annotation.

### Usage

```
concatenate_textgrids(path, result_file_name = "concatenated")
```

### Arguments

path	path to the directory with soundfiles.
result_file_name	name of the result and annotation files.

### Value

no output

### Author(s)

George Moroz <agricolamz@gmail.com>

**Examples**

```
# create two files in a temporary folder "test_folder"
t1 <- system.file("extdata", "test.TextGrid", package = "phonfieldwork")
t2 <- system.file("extdata", "post.TextGrid", package = "phonfieldwork")
tdir <- tempdir()
file.copy(c(t1, t2), tdir)

# here are two .wav files in a folder
list.files(tdir)
# [1] "post.TextGrid" "test.TextGrid" ...

# Concatenate all TextGrids from the folder into concatenated.TextGrid
concatenate_textgrids(path = tdir, result_file_name = "concatenated")

list.files(tdir)
# [1] "concatenated.TextGrid" "post.TextGrid" "test.TextGrid" ...
```

---

create\_empty\_textgrid *Create an empty TextGrid*

---

**Description**

Creates an empty Praat TextGrid in the same folder as a reference sound file. It is possible to manage with predefined number of tiers, their names and their types.

**Usage**

```
create_empty_textgrid(
  duration,
  tiers = NULL,
  point_tiers = NULL,
  path,
  result_file_name = "new_textgrid"
)
```

**Arguments**

duration	integer. Duration of the textgrid. If you do not know the duration of your audio file use the <code>get_sound_duration()</code> function.
tiers	a vector that contain tier names.
point_tiers	a vector that defines which tiers should be made point tiers. This argument accepts numeric values (e. g. <code>c(2,4)</code> means second and forth tiers) or character (e. g. <code>c("a", "b")</code> means tiers with names "a" and "b")
path	path to the directory with soundfiles.
result_file_name	name of the result and annotation files.

**Value**

The function returns no output, just creates a Praat TextGrid in the same folder as a reference sound file.

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

```
create_glossed_document
      Create a glossed document
```

---

**Description**

Creates a file with glossed example (export from .flextext or other formats)

**Usage**

```
create_glossed_document(
  flextext = NULL,
  rows = c("gls"),
  output_dir,
  output_file = "glossed_document",
  output_format = "html",
  example_pkg = NULL
)
```

**Arguments**

flextext	path to a .flextext file or a dataframe with the following columns: p_id, s_id, w_id, txt, cf, hn, gls, msa, morph, word, phrase, paragraph, free_trans, text, text_title
rows	vector of row names from the flextext that should appear in the final document. Possible values are: "cf", "hn", "gls", "msa". "gls" is default.
output_dir	the output directory for the rendered file
output_file	the name of the result .html file (by default glossed_document).
output_format	The option can be "html" or "docx"
example_pkg	vector with name of the LaTeX package for glossing (possible values: "gb4e", "langsci", "expex", "philex")

**Value**

If render is FALSE, the function returns a path to the temporary file with .csv file. If render is TRUE, there is no output in a function.



**Author(s)**

George Moroz <agricolamz@gmail.com>

---

create\_image\_look\_up    *Create image look\_up objects for html viewer*

---

**Description**

Create image look\_up objects for html viewer

**Usage**

```
create_image_look_up(img_src, img_caption = NULL, text = "&#x1f441;")
```

**Arguments**

img_src	string or vector of strings with a image(s) path(s).
img_caption	string or vector of strings that will be displayed when image is clicked.
text	string o vector of strings that will be displayed as view link. By default it is eye emoji (&#x1f441;).

**Value**

a string or vector of strings

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

create\_presentation    *Creates a presentation*

---

**Description**

Creates an html or powerpoint presentation in a working directory from list of words and translations. [Here](#) is an example of such presentation.

**Usage**

```
create_presentation(  
  stimuli,  
  translations = "",  
  external = NULL,  
  font_size = 50,  
  output_dir,  
  output_format = "html",  
  output_file = "stimuli_presentation",  
  render = TRUE  
)
```

**Arguments**

stimuli	the vector of stimuli (obligatory). Can be a path to an image.
translations	the vector of translations (optional)
external	the vector with the indices of external images
font_size	font size in px (50, by default)
output_dir	the output directory for the rendered file
output_format	the string that define the R Markdown output format: "html" (by default) or "pptx"
output_file	the name of the result presentation file (by default stimuli_presentation)
render	the logical argument, if TRUE render the created R Markdown presentation to the output_dir folder, otherwise returns the path to the temporary file with a Rmd file.

**Value**

If render is FALSE, the function returns a path to the temporary file. If render is TRUE, there is no output in a function.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
create_presentation(  
  stimuli = c("rzeka", "drzewo"),  
  translations = c("river", "tree"),  
  render = FALSE  
)  
  
# with image  
create_presentation(  
  stimuli = c(  
    "rzeka", "drzewo",  
    system.file("extdata", "r-logo.png"),
```

```

        package = "phonfieldwork"
    )
),
translations = c("river", "tree", ""),
external = 3,
render = FALSE
)

```

---

create\_sound\_play      *Create audio play objects for html viewer*

---

### Description

Create audio play objects for html viewer

### Usage

```
create_sound_play(snd_src, text = "&#x1f442;")
```

### Arguments

snd_src	string or vector of strings with a image(s) path(s).
text	string o vector of strings that will be displayed as view link. By default it is ear emoji (&#x1f442;).

### Value

a string or vector of strings

### Author(s)

George Moroz <agricolamz@gmail.com>

---

create\_subannotation      *Create boundaries in a texgrid tier*

---

### Description

Create boundaries in a texgrid tier

**Usage**

```
create_subannotation(
  textgrid,
  tier = 1,
  new_tier_name = "",
  n_of_annotations = 4,
  each = 1,
  omit_blank = TRUE,
  overwrite = TRUE
)
```

**Arguments**

textgrid	character with a filename or path to the TextGrid
tier	value that could be either ordinal number of the tier either name of the tier
new_tier_name	a name of a new created tier
n_of_annotations	number of new annotations per annotation to create
each	non-negative integer. Each new blank annotation is repeated every first, second or ... times
omit_blank	logical. If TRUE (by default) it doesn't create subannotation for empty annotations.
overwrite	logical. If TRUE (by default) it overwrites an existing tier.

**Value**

a string that contain TextGrid. If argument write is TRUE, then no output.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
create_subannotation(system.file("extdata", "test.TextGrid",
  package = "phonfieldwork"
),
  tier = 1, overwrite = FALSE
)
```

---

create_viewer	<i>Create an annotation viewer</i>
---------------	------------------------------------

---

## Description

Creates an html file with table and sound preview and player

## Usage

```
create_viewer(
  audio_dir,
  picture_dir,
  table,
  captions = NULL,
  sorting_columns = NULL,
  about = "Created with the `phonfieldworks` package (Moroz 2020).",
  map = FALSE,
  output_dir,
  output_file = "stimuli_viewer",
  render = TRUE
)
```

## Arguments

audio_dir	path to the directory with sounds
picture_dir	path to the directory with pictures
table	data frame with data ordered according to files in the audio folder
captions	vector of strings that will be used for captions for a picture.
sorting_columns	vector of strings for sorting the result column
about	it is either .Rmd file or string with the text for about information: author, project, place of gathered information and other metadata, version of the viewer and so on
map	the logical argument, if TRUE and there is a glottocode column in table
output_dir	the output directory for the rendered file
output_file	the name of the result .html file (by default stimuli_viewer)
render	the logical argument, if TRUE renders the created R Markdown viewer to the output_dir folder, otherwise returns the path to the temporary file with a .csv file.

## Value

If render is FALSE, the function returns a path to the temporary file with .csv file. If render is TRUE, there is no output in a function.

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

df\_to\_tier

*Dataframe to TextGrid's tier*

---

**Description**

Convert a dataframe to a Praat TextGrid.

**Usage**

```
df_to_tier(df, textgrid, tier_name = "", overwrite = TRUE)
```

**Arguments**

df	an R dataframe object that contains columns named "content", "time_start" and "time_end"
textgrid	a character with a filename or path to the TextGrid
tier_name	a vector that contain a name for a created tier
overwrite	a logic argument, if TRUE overwrites the existing TextGrid file

**Value**

If overwrite is FALSE, then the function returns a vector of strings with a TextGrid. If overwrite is TRUE, then no output.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
time_start <- c(0.00000000, 0.01246583, 0.24781914, 0.39552363, 0.51157715)
time_end <- c(0.01246583, 0.24781914, 0.39552363, 0.51157715, 0.65267574)
content <- c("", "T", "E", "S", "T")
df_to_tier(my_df <- data.frame(id = 1:5, time_start, time_end, content),
  system.file("extdata", "test.TextGrid",
    package = "phonfieldwork"
  ),
  overwrite = FALSE
)
```

---

`draw_sound`*Draw Oscilogram, Spectrogram and annotation*

---

**Description**

Create oscilogram and spectrogram plot.

**Usage**

```
draw_sound(  
  file_name,  
  annotation = NULL,  
  from = NULL,  
  to = NULL,  
  zoom = NULL,  
  text_size = 1,  
  output_file = NULL,  
  title = NULL,  
  freq_scale = "kHz",  
  frequency_range = c(0, 5),  
  dynamic_range = 50,  
  window_length = 5,  
  window = "kaiser",  
  windowparameter = -1,  
  preemphasisf = 50,  
  spectrum_info = TRUE,  
  raven_annotation = NULL,  
  formant_df = NULL,  
  pitch = NULL,  
  pitch_range = c(75, 350),  
  intensity = NULL,  
  output_width = 750,  
  output_height = 500,  
  output_units = "px",  
  sounds_from_folder = NULL,  
  textgrids_from_folder = NULL,  
  pic_folder_name = "pics",  
  title_as_filename = TRUE,  
  prefix = NULL,  
  suffix = NULL,  
  autonumber = FALSE  
)
```

**Arguments**

`file_name`      a sound file

annotation	a source for annotation files (path to TextGrid file or dataframe created from other linguistic types, e. g. via <code>textgrid_to_df()</code> , <code>eaf_to_df()</code> or other functions)
from	Time in seconds at which to start extraction.
to	Time in seconds at which to stop extraction.
zoom	numeric vector of zoom window time (in seconds). It will draw the whole oscilogram and part of the spectrogram.
text_size	numeric, text size (default = 1).
output_file	the name of the output file
title	the title for the plot
freq_scale	a string indicating the type of frequency scale. Supported types are: "Hz" and "kHz".
frequency_range	vector with the range of frequencies to be displayed for the spectrogram up to a maximum of $fs/2$ . By default this is set to 0-5 kHz.
dynamic_range	values greater than this many dB below the maximum will be displayed in the same color
window_length	the desired analysis window length in milliseconds.
window	A string indicating the type of window desired. Supported types are: "rectangular", "hann", "hamming", "cosine", "bartlett", "gaussian", and "kaiser".
windowparameter	The parameter necessary to generate the window, if appropriate. At the moment, the only windows that require parameters are the Kaiser and Gaussian windows. By default, these are set to 2 for kaiser and 0.4 for gaussian windows.
preemphasisf	Preemphasis of 6 dB per octave is added to frequencies above the specified frequency. For no preemphasis, set to a frequency higher than the sampling frequency.
spectrum_info	logical. If TRUE then add information about window method and params.
raven_annotation	Raven (Center for Conservation Bioacoustics) style annotations (boxes over spectrogram). The dataframe that contains <code>time_start</code> , <code>time_end</code> , <code>freq_low</code> and <code>freq_high</code> columns. Optional columns are <code>colors</code> and <code>content</code> .
formant_df	dataframe with formants from <code>formant_to_df()</code> function
pitch	path to the Praat '.Pitch' file or result of <code>pitch_to_df()</code> function. This variable provide data for visualisation of a pitch contour exported from Praat.
pitch_range	vector with the range of frequencies to be displayed. By default this is set to 75-350 Hz.
intensity	path to the Praat '.Intensity' file or result of <code>intensity_to_df()</code> function. This variable provide data for visualisation of an intensity contour exported from Praat.
output_width	the width of the device
output_height	the height of the device



output_units	the units in which height and width are given. Can be "px" (pixels, the default), "in" (inches), "cm" or "mm".
sounds_from_folder	path to a folder with multiple sound files. If this argument is not NULL, then the function goes through all files and creates picture for all of them.
textgrids_from_folder	path to a folder with multiple .TextGrid files. If this argument is not NULL, then the function goes through all files and create picture for all of them.
pic_folder_name	name for a folder, where all pictures will be stored in case sounds_from_folder argument is not NULL
title_as_filename	logical. If true adds filename title to each picture
prefix	prefix for all file names for created pictures in case sounds_from_folder argument is not NULL
suffix	suffix for all file names for created pictures in case sounds_from_folder argument is not NULL
autonumber	if TRUE automatically add number of extracted sound to the file_name. Prevents from creating a duplicated files and wrong sorting.

**Value**

Oscilogram and spectrogram plot (and possibly TextGrid annotation).

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
## Not run:
draw_sound(system.file("extdata", "test.wav", package = "phonfieldwork"))

draw_sound(
  system.file("extdata", "test.wav", package = "phonfieldwork"),
  system.file("extdata", "test.TextGrid",
    package = "phonfieldwork"
  )
)

draw_sound(system.file("extdata", "test.wav", package = "phonfieldwork"),
  system.file("extdata", "test.TextGrid", package = "phonfieldwork"),
  pitch = system.file("extdata", "test.Pitch",
    package = "phonfieldwork"
  ),
  pitch_range = c(50, 200)
)
draw_sound(system.file("extdata", "test.wav", package = "phonfieldwork"),
  system.file("extdata", "test.TextGrid", package = "phonfieldwork"),
```

```
pitch = system.file("extdata", "test.Pitch",
  package = "phonfieldwork"
),
pitch_range = c(50, 200),
intensity = intensity_to_df(system.file("extdata", "test.Intensity",
  package = "phonfieldwork"
))
)
draw_sound(system.file("extdata", "test.wav", package = "phonfieldwork"),
  formant_df = formant_to_df(system.file("extdata", "e.Formant",
  package = "phonfieldwork"
))
)

## End(Not run)
```

---

draw\_spectrogram      *Draw spectrograms*

---

## Description

This function was slightly changed from `phonTools::spectrogram()`. Argument description is copied from `phonTools::spectrogram()`.

## Usage

```
draw_spectrogram(
  sound,
  fs = 22050,
  text_size = 1,
  window_length = 5,
  dynamic_range = 50,
  window = "kaiser",
  windowparameter = -1,
  freq_scale = "kHz",
  spectrum_info = TRUE,
  timestep = -1000,
  padding = 10,
  preemphasisf = 50,
  frequency_range = c(0, 5),
  nlevels = dynamic_range,
  x_axis = TRUE,
  title = NULL,
  raven_annotation = NULL,
  formant_df = NULL
)
```

**Arguments**

sound	Either a numeric vector representing a sequence of samples taken from a sound wave or a sound object created with the <code>loadsound()</code> or <code>makesound()</code> functions.
fs	The sampling frequency in Hz. If a sound object is passed this does not need to be specified.
text_size	numeric, text size (default = 1).
window_length	The desired analysis window length in milliseconds.
dynamic_range	Values greater than this many dB below the maximum will be displayed in the same color.
window	A string indicating the type of window desired. Supported types are: rectangular, hann, hamming, cosine, bartlett, gaussian, and kaiser.
windowparameter	The parameter necessary to generate the window, if appropriate. At the moment, the only windows that require parameters are the Kaiser and Gaussian windows. By default, these are set to 2 for kaiser and 0.4 for gaussian windows.
freq_scale	a string indicating the type of frequency scale. Supported types are: "Hz" and "kHz".
spectrum_info	logical. If TRUE then add information about window method and params.
timestep	If a negative value is given, -N, then N equally-spaced time steps are calculated. If a positive number is given, this is the spacing between adjacent analyses, in milliseconds.
padding	The amount of zero padding for each window, measured in units of window length. For example, if the window is 50 points, and padding = 10, 500 zeros will be appended to each window.
preemphasisf	Preemphasis of 6 dB per octave is added to frequencies above the specified frequency. For no preemphasis, set to a frequency higher than the sampling frequency.
frequency_range	vector with the range of frequencies to be displayed for the spectrogram up to a maximum of $fs/2$ . This is set to 0-5 kHz by default.
nlevels	The number of divisions to be used for the z-axis of the spectrogram. By default it is set equal to the dynamic range, meaning that a single color represents 1 dB on the z-axis.
x_axis	If TRUE then draw x axis.
title	Character with the title.
raven_annotation	Raven (Center for Conservation Bioacoustics) style annotations (boxes over spectrogram). The dataframe that contains <code>time_start</code> , <code>time_end</code> , <code>freq_low</code> and <code>freq_high</code> columns. Optional columns are <code>colors</code> and <code>content</code> .
formant_df	dataframe with formants from <code>formant_to_df()</code> function

**Author(s)**

Santiago Barreda <sbarreda@ucdavis.edu>

**Examples**

```
## Not run:
draw_spectrogram(system.file("extdata", "test.wav",
  package = "phonfieldwork"
))

## End(Not run)
```

---

eaf\_to\_df

*ELAN's .eaf file to dataframe*

---

**Description**

Convert .eaf file from ELAN to a dataframe.

**Usage**

```
eaf_to_df(file_name)
```

**Arguments**

file\_name        string with a filename or path to the .eaf file

**Value**

a dataframe with columns: tier, id, content, tier\_name, tier\_type, time\_start, time\_end, source).

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
eaf_to_df(system.file("extdata", "test.eaf", package = "phonfieldwork"))
```

---

exb_to_df	<i>EXMARaLDA's .exb file to dataframe</i>
-----------	---

---

**Description**

Convert .exb file from EXMARaLDA to a dataframe.

**Usage**

```
exb_to_df(file_name)
```

**Arguments**

file\_name        string with a filename or path to the .exb file

**Value**

a dataframe with columns: tier, id, content, tier\_name, tier\_type, tier\_category, tier\_speaker, time\_start, time\_end, source.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
exb_to_df(system.file("extdata", "test.exb", package = "phonfieldwork"))
```

---

extract_intervals	<i>Extract intervals</i>
-------------------	--------------------------

---

**Description**

Extract sound according to non-empty annotated intervals from TextGrid and create soundfiles with correspondent names.

**Usage**

```
extract_intervals(  
  file_name,  
  textgrid,  
  tier = 1,  
  prefix = NULL,  
  suffix = NULL,  
  autonumber = TRUE,  
  path  
)
```

**Arguments**

file_name	path to the soundfile
textgrid	path to the TextGrid
tier	tier number or name that should be used as base for extraction and names
prefix	character vector containing prefix(es) for file names
suffix	character vector containing suffix(es) for file names
autonumber	if TRUE automatically add number of extracted sound to the file_name. Prevents from creating a duplicated files and wrong sorting.
path	path to the directory where create extracted soundfiles.

**Value**

no output

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
# create two files in a temprary folder "test_folder"
s <- system.file("extdata", "test.wav", package = "phonfieldwork")
tdir <- tempdir()
file.copy(s, tdir)

# Extract intervals according the TextGrid into the path
extract_intervals(
  file_name = paste0(tdir, "/test.wav"),
  textgrid = system.file("extdata", "test.TextGrid",
    package = "phonfieldwork"
  ),
  path = tdir
)

list.files(tdir)
# [1] "e-2.wav" "s-3.wav" "t-1.wav" "t-4.wav" "test.wav"
```

---

flextext\_to\_df

*FLEX's .flextext file to dataframe*


---

**Description**

Convert .flextext file from FLEX to dataframe.

**Usage**

```
flextext_to_df(file_name)
```

**Arguments**

file\_name        string with a filename or path to the .flextext file

**Value**

a dataframe with columns: p\_id, s\_id, w\_id, txt, cf, hn, gls, msa, morph, word, phrase, paragraph, free\_trans, text, text\_title

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

formant_to_df	<i>Praat Formant object to dataframe</i>
---------------	--

---

**Description**

Convert a Praat Formant object to a dataframe.

**Usage**

```
formant_to_df(file_name)
```

**Arguments**

file\_name        string with a filename or path to the Formant file

**Value**

a dataframe with columns: time\_start, time\_end, frequency, bandwidth and formant

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
formant_to_df(system.file("extdata", "e.Formant", package = "phonfieldwork"))
```

get\_sound\_duration      *Get file(s) duration*

---

**Description**

Calculate sound(s) duration.

**Usage**

```
get_sound_duration(file_name)
```

**Arguments**

file\_name      a sound file

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
get_sound_duration(  
  system.file("extdata", "test.wav", package = "phonfieldwork")  
)
```

---

get\_textgrid\_names      *Extract TextGrid names*

---

**Description**

Extract TextGrid names.

**Usage**

```
get_textgrid_names(textgrid)
```

**Arguments**

textgrid      path to the TextGrid

**Value**

return a vector of tier names from given TextGrid

**Author(s)**

George Moroz <agricolamz@gmail.com>



**Examples**

```
get_textgrid_names(system.file("extdata", "test.TextGrid",
  package = "phonfieldwork"
))
```

---

intensity\_to\_df      *Praat Intensity tier to dataframe*

---

**Description**

Convert a Praat Intensity tier to a dataframe.

**Usage**

```
intensity_to_df(file_name)
```

**Arguments**

file\_name      string with a filename or path to the Intensity tier

**Value**

a dataframe with columns: time\_start, time\_end, Intensity

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
intensity_to_df(system.file("extdata", "test.Intensity", package = "phonfieldwork"))
```

---

pitch\_to\_df      *Praat Pitch tier to dataframe*

---

**Description**

Convert a Praat Pitch tier to a dataframe.

**Usage**

```
pitch_to_df(file_name, candidates = "")
```

**Arguments**

file_name	string with a filename or path to the Pitch tier
candidates	Praat Pitch tier contains multiple candidates for each time slice, use the value "all" if you want to get them all

**Value**

a dataframe with columns: time\_start, time\_end, frequency and, if candidates = "all", candidate\_id and strength

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
pitch_to_df(system.file("extdata", "test.Pitch", package = "phonfieldwork"))
```

---

read_from_folder	<i>Read multiple files from the folder</i>
------------------	--

---

**Description**

This function reads multiple files from the folder. The first argument is the path, the second argument is the type of files to read.

**Usage**

```
read_from_folder(path, type)
```

**Arguments**

path	to a folder with multiple sound files.
type	should be one of the following: "duration", "audacity", "eaf", "exb", "flextext", "formant", "intensity", "pitch", "srt", "textgrid"

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
read_from_folder(system.file("extdata", package = "phonfieldwork"), "eaf")
```

---

rename_soundfiles	<i>Rename soundfiles</i>
-------------------	--------------------------

---

### Description

Rename soundfiles using the template from user.

### Usage

```
rename_soundfiles(  
  stimuli,  
  translations = NULL,  
  prefix = NULL,  
  suffix = NULL,  
  order = NULL,  
  missing = NULL,  
  path,  
  autonumbering = TRUE,  
  backup = TRUE,  
  logging = TRUE  
)
```

### Arguments

stimuli	character vector of stimuli
translations	character vector of translations (optional). This values are added after stimuli to the new files' names so the result will be ...stimulus_translation....
prefix	character vector of length one containing prefix for file names
suffix	character vector of length one containing suffix for file names
order	numeric vector that define the order of stimuli. By default the order of the stimuli is taken.
missing	numeric vector that define missing stimuli in case when some stimuli are not recorded.
path	path to the directory with soundfiles.
autonumbering	logical. If TRUE, function creates an automatic numbering of files.
backup	logical. If TRUE, function creates backup folder with all files. By default is TRUE.
logging	logical. If TRUE creates a .csv file with the correspondences of old names and new names. This could be useful for restoring in case something goes wrong.

### Value

no output

**Author(s)**

George Moroz <agricolamz@gmail.com>

---

set\_textgrid\_names      *Rewrite TextGrid names*

---

**Description**

Rewrite TextGrid names.

**Usage**

```
set_textgrid_names(textgrid, tiers, names, write = TRUE)
```

**Arguments**

textgrid	path to the TextGrid
tiers	integer vector with the number of tiers that should be named
names	vector of strings with new names for TextGrid tiers
write	logical. If TRUE (by default) it overwrites an existing tier

**Value**

a string that contain TextGrid. If argument write is TRUE, then no output.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
set_textgrid_names(system.file("extdata", "test.TextGrid",  
  package = "phonfieldwork"  
),  
tiers = 3, names = "new_name", write = FALSE  
)
```

---

srt_to_df	<i>Subtitles .srt file to dataframe</i>
-----------	---

---

**Description**

Convert subtitles .srt file to a dataframe.

**Usage**

```
srt_to_df(file_name)
```

**Arguments**

file\_name      string with a filename or path to the .srt file

**Value**

a dataframe with columns: id, content, time\_start, time\_end, source.

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
srt_to_df(system.file("extdata", "test.srt", package = "phonfieldwork"))
```

---

textgrid_to_df	<i>TextGrid to dataframe</i>
----------------	------------------------------

---

**Description**

Convert Praat TextGrid to a dataframe.

**Usage**

```
textgrid_to_df(file_name)
```

**Arguments**

file\_name      string with a filename or path to the TextGrid

**Value**

a dataframe with columns: id, time\_start, time\_end (if it is an interval tier – the same as the start value), content, tier, tier\_name and source

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
textgrid_to_df(system.file("extdata", "test.TextGrid",
  package = "phonfieldwork"
))

# this is an example of reading a short .TextGrid format
textgrid_to_df(system.file("extdata", "test_short.TextGrid",
  package = "phonfieldwork"
))
```

---

tier_to_df	<i>TextGrid's tier to dataframe</i>
------------	-------------------------------------

---

**Description**

Convert selected tier from a Praat TextGrid to a dataframe.

**Usage**

```
tier_to_df(file_name, tier = 1)
```

**Arguments**

file_name	string with a filename or path to the TextGrid
tier	value that could be either ordinal number of the tier either name of the tier. By default is '1'.

**Value**

a dataframe with columns: id, time\_start, time\_end, content, , tier\_name

**Author(s)**

George Moroz <agricolamz@gmail.com>

**Examples**

```
tier_to_df(system.file("extdata", "test.TextGrid",
  package = "phonfieldwork"
))
tier_to_df(
  system.file("extdata", "test.TextGrid",
  package = "phonfieldwork"
),
"intervals"
)
```

# Index

[add\\_leading\\_symbols](#), 3  
[annotate\\_textgrid](#), 3  
[audacity\\_to\\_df](#), 4

[concatenate\\_soundfiles](#), 5  
[concatenate\\_textgrids](#), 6  
[create\\_empty\\_textgrid](#), 7  
[create\\_glossed\\_document](#), 8  
[create\\_image\\_look\\_up](#), 9  
[create\\_presentation](#), 9  
[create\\_sound\\_play](#), 11  
[create\\_subannotation](#), 11  
[create\\_viewer](#), 13

[df\\_to\\_tier](#), 14  
[draw\\_sound](#), 15  
[draw\\_spectrogram](#), 18

[eaf\\_to\\_df](#), 20  
[exb\\_to\\_df](#), 21  
[extract\\_intervals](#), 21

[flextext\\_to\\_df](#), 22  
[formant\\_to\\_df](#), 23

[get\\_sound\\_duration](#), 24  
[get\\_textgrid\\_names](#), 24

[intensity\\_to\\_df](#), 25

[pitch\\_to\\_df](#), 25

[read\\_from\\_folder](#), 26  
[rename\\_soundfiles](#), 27

[set\\_textgrid\\_names](#), 28  
[srt\\_to\\_df](#), 29

[textgrid\\_to\\_df](#), 29  
[tier\\_to\\_df](#), 30