

Package ‘glossr’

July 22, 2025

Type Package

Title Use Interlinear Glosses in R Markdown

Version 0.8.0

Description Read examples with interlinear glosses from files
or from text and print them in a way compatible with both
Latex and HTML outputs.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports tibble, dplyr, knitr, purrr, rlang, cli, yaml, systemfonts

RoxygenNote 7.3.1

Config/testthat/edition 3

VignetteBuilder knitr

Suggests bookdown, testthat, rmarkdown, stringr, htmltools, covr

Depends R (>= 4.1)

URL <https://montesmariana.github.io/glossr/>,
<https://github.com/montesmariana/glossr>

Language en-GB

NeedsCompilation no

Author Mariana Montes [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-3869-3207>>),
Benjamin Chauvette [cph] (Author of included leipzig.js library)

Maintainer Mariana Montes <montesmariana@gmail.com>

Repository CRAN

Date/Publication 2024-05-19 13:30:03 UTC

Contents

as_gloss	2
config_from_file	3
gloss	4
glosses	4
gloss_data	5
gloss_df	6
gloss_factory	7
gloss_format_words	8
gloss_linetooltip	8
gloss_list	9
gloss_render	10
knit_print.gloss	11
new_gloss	11
print_config	12
set_style_options	12
use_glossr	14
Index	15

as_gloss	<i>Helper to create gloss objects</i>
----------	---------------------------------------

Description

Based on a character vectors and up to three label arguments, create an object where those arguments are attributes. These are:

- **source**: Where the text comes from. This will be printed in the first line of the example, without word alignment.
- **translation**: Free translation. This will be printed as the last line of the example, without word alignment and in quotation marks if so desired.
- **label**: Named label of the example, for cross-references.
- **lengths**: This is computed within the function, not provider, and it's the number of items identified in each gloss line.

Usage

```
as_gloss(
  ...,
  source = NULL,
  translation = NULL,
  label = NULL,
  trans_quotes = config$trans_quotes,
  output_format = config$output,
  numbering = config$numbering
)
```

Arguments

...	Lines for glossing
source	(Optional) Source of example
translation	(Optional) Free translation
label	(Optional) Example label
trans_quotes	(Optional) Quotes to surround the free translation with.
output_format	(Optional) Whether it will use latex, word or html format.
numbering	(Optional) Whether the gloss should be numbered (in Word and HTML).

Value

Object of class `gloss`, ready to be printed based on the chosen output format, and with a `gloss_data` object as data attribute (or, in the case of calls via `gloss_df()`, the original input asdata).

Examples

```
ex_sp <- "Un ejemplo en español"
ex_gloss <- "DET.M.SG example in Spanish"
ex_trans <- "An example in Spanish"
my_gloss <- as_gloss(ex_sp, ex_gloss, translation = ex_trans, label="ex1")

# check the gloss data
attr(my_gloss, "data")
```

config_from_file	<i>Override configuration with a YAML file</i>
------------------	--

Description

Read configuration from a YAML file to provide instructions for styling and PDF- or Word-specific options.

Usage

```
config_from_file(filename)
```

Arguments

filename	Path to the YAML configuration file, e.g. "glossr-config.yaml".
----------	---

Value

Invisibly, the contents of the configuration file that passed validation.

Examples

```
config_file <- system.file("extdata/glossr-config.yml", package="glossr")
config_from_file(config_file)
print_config()
```

gloss	<i>Reference gloss</i>
-------	------------------------

Description

Latex output uses `\@ref(label)` to reference examples, whereas HTML output is based on pandoc examples, i.e. `(@label)`. ``r gloss(label)``, written inline in the text, will return the appropriate reference based on the selected output.

Usage

```
gloss(label)
```

Arguments

label	Label for reference
-------	---------------------

Value

Character string with label reference

glosses	<i>Examples of glosses</i>
---------	----------------------------

Description

A dataset containing five glossing examples extracted from Koptjevskaja-Tamm (2015)'s *The Linguistics of Temperature* and chapters within.

Usage

```
glosses
```

Format

A `tibble::tibble` with 5 rows and 6 variables:

original The text in the original language.

parsed The text with translations to English or morphological annotation per word or expression, with LaTeX formatting.

translation Free translation to English.

label Label for referencing the example.

language Original language of the text.

Source Where the example was taken from (published paper).

gloss_data

gloss_data *class*

Description

Based on a character vectors and up to three label arguments, create an object where those arguments are attributes. These are:

- **source**: Where the text comes from. This will be printed in the first line of the example, without word alignment.
- **translation**: Free translation. This will be printed as the last line of the example, without word alignment and in quotation marks if so desired.
- **label**: Named label of the example, for cross-references.
- **lengths**: This is computed within the function, not provider, and it's the number of items identified in each gloss line.

Usage

```
new_gloss_data(
  gloss_lines,
  source = NULL,
  translation = NULL,
  label = NULL,
  trans_quotes = config$trans_quotes
)
```

Arguments

<code>gloss_lines</code>	Lines for glossing, as a list
<code>source</code>	(Optional) Source of example
<code>translation</code>	(Optional) Free translation
<code>label</code>	(Optional) Example label
<code>trans_quotes</code>	(Optional) Quotes to surround the free translation with.

Details

This function is mostly for internal use, but may be useful for debugging or checking the output of specific calls. Normally, it's best to use `as_gloss()` or `gloss_df()`. Note that, unlike `as_gloss()`, `new_gloss_data` requires a list of gloss lines.

Value

Object of class `gloss_data`

<code>gloss_df</code>	<i>Render gloss from a dataframe</i>
-----------------------	--------------------------------------

Description

Render gloss from a dataframe

Usage

```
gloss_df(df, output_format = config$output, numbering = config$numbering)
```

Arguments

<code>df</code>	Dataframe one row per gloss. Columns <code>translation</code> , <code>source</code> and <code>label</code> have special meaning (see <code>as_gloss()</code>); all the others will be interpreted as lines to align in the order given.
<code>output_format</code>	(Optional) Whether it will use latex, word or html format.
<code>numbering</code>	(Optional) Whether the gloss should be numbered (in Word and HTML).

Value

Object of class `gloss` with the original input as `data` attribute.

Examples

```
my_gloss <- data.frame(
  first_line = "my first line",
  second_line = "my second line",
  translation = "Translation of my example",
  label = "label"
)
gloss_df(my_gloss)
```

gloss_factory	<i>Function factory to print glosses from dataframe</i>
---------------	---

Description

This function takes a dataframe with glosses and returns another function that takes either an id or list of ids (if `use_conditionals` is FALSE) or a conditional statement (if TRUE) and runs `gloss_df()` on the filtered dataframe.

Usage

```
gloss_factory(  
  glosses,  
  use_conditionals = FALSE,  
  id_column = "label",  
  ignore_columns = NULL,  
  validate = TRUE  
)
```

Arguments

<code>glosses</code>	Dataframe with gloss data.
<code>use_conditionals</code>	Boolean. If TRUE, the returned function will use conditional statements to filter the dataframe. Otherwise, it will use ids and match them to the values in the <code>id_column</code> .
<code>id_column</code>	Name of the column with ids for filtering, if <code>use_conditionals</code> is FALSE.
<code>ignore_columns</code>	Optional character vector with names of columns that could be used for filtering but should not be provided to <code>gloss_df()</code> .
<code>validate</code>	Boolean. If TRUE, running <code>gloss_factory()</code> will print a few informative messages about how glossr is reading the dataframe.

Value

A function.

If `use_conditionals` is FALSE (the default), the returned function will take a character vector or a series of character vectors with id's to filter. If `id_column` is "label", running that function will be the equivalent to filtering glosses based on the values in the `label` column.

If `use_conditionals` is TRUE, the returned function will take the same conditions that a `dplyr::filter()` would.

Examples

```
my_glosses <- dplyr::select(glosses, -language)
by_label <- gloss_factory(my_glosses)

by_label("heartwarming-jp")

by_label("heartwarming-jp", "languid-jp")

by_cond <- gloss_factory(my_glosses, use_conditional = TRUE)
by_cond(stringr::str_ends(label, "jp"))
```

gloss_format_words	<i>Apply latex formatting to many words</i>
--------------------	---

Description

Facilitates applying the same latex formatting to different words in a row.

Usage

```
gloss_format_words(text, formatting)
```

Arguments

text	Character vector of length 1.
formatting	Latex formatting code, e.g. textit or textsc.

Value

Reformatted string

Examples

```
gloss_format_words("Many words to apply italics on.", "textit")
```

gloss_linetooltip	<i>Apply tooltip to a full gloss</i>
-------------------	--------------------------------------

Description

Apply tooltip to a full gloss

Usage

```
gloss_linetooltip(original, parsed)
```

Arguments

original	Text to show in the tooltip rendering.
parsed	Text to show as tooltip when hovering

Value

List of [shiny.tag](#)

Examples

```
ex_sp <- "Un ejemplo en español"
ex_gloss <- "DET.M.SG example in Spanish"
gloss_linetooltip(ex_sp, ex_gloss)
```

gloss_list	<i>Sublist glosses</i>
------------	------------------------

Description

Takes a series of glosses from [gloss_render\(\)](#) and puts them in a list within one example for PDF output.

Usage

```
gloss_list(glist, listlabel = NULL)
```

Arguments

glist	Concatenation of gloss objects, e.g. as output of gloss_df() .
listlabel	Label for the full list (optional)

Value

Character vector including the frame for a list of glosses.

gloss_render	<i>Render a gloss</i>
--------------	-----------------------

Description

These functions are output-specific and can be used to check the specific output of certain calls, but are not meant to be used in an R Markdown file. Instead, use [as_gloss\(\)](#) or [gloss_df\(\)](#).

Usage

```
gloss_pdf(gloss)

gloss_html(gloss, numbering = TRUE)

gloss_tooltip(gloss, numbering = TRUE)

gloss_leipzig(gloss, numbering = TRUE)

gloss_word(gloss, numbering = TRUE)
```

Arguments

gloss	Object of class gloss_data
numbering	Whether the gloss should be numbered (in HTML and Word).

Value

Object of class [gloss](#).

Functions

- `gloss_pdf()`: Render in PDF
- `gloss_html()`: Render in HTML
- `gloss_tooltip()`: Tooltip rendering for HTML
- `gloss_leipzig()`: Leipzig.js engine
- `gloss_word()`: Render in Word

Examples

```
ex_sp <- "Un ejemplo en español"
ex_gloss <- "DET.M.SG example in Spanish"
ex_trans <- "An example in Spanish"
my_gloss <- new_gloss_data(list(ex_sp, ex_gloss), translation = ex_trans, label="ex1")
gloss_pdf(my_gloss)

gloss_html(my_gloss)
```

knit_print.gloss	<i>Print method for glosses</i>
------------------	---------------------------------

Description

This method print [gloss](#) objects with {knitr}.

Usage

```
## S3 method for class 'gloss'
knit_print(x, ...)
```

Arguments

x	Object to print
...	Other options

new_gloss	<i>gloss class</i>
-----------	--------------------

Description

The `gloss` class contains how a gloss will be printed and its original input (Object of class [gloss_data](#)) as data attribute. It also has a `knitr::knit_print()` method for rendering in R Markdown (and Quarto).

Usage

```
new_gloss(input, output)
```

Arguments

input	A gloss_data object.
output	How the gloss must be printed, depending on the output.

Value

Object of class `gloss`.

print_config	<i>Print the glossr configuration</i>
--------------	---------------------------------------

Description

Print the glossr configuration

Usage

```
print_config(config_sections = c("format", "pdf", "word", "other"))
```

Arguments

config_sections

Character vector with the following possible values:

- **format:** to show the formatting options (italics / bold / nothing) for the different lines.
- **pdf:** to show LaTeX-specific formatting
- **word:** to show Word-specific formatting
- **other:** to show the current output format, translation quotes and the numbering setting.

Value

Invisibly, a named list with the printed values

Examples

```
print_config()
print_config("pdf") # print only pdf-specific configuration
```

set_style_options	<i>Set general styling options</i>
-------------------	------------------------------------

Description

This is a helper function to set the configuration that control style characteristics for glosses across the full document. It is called within [use_glossr\(\)](#).

Usage

```
set_style_options(styling = list())
```

Arguments

styling Named list of styling options for specific elements of glosses.

Details

There are five types of settings that can be provided in the list.

First, `trans_quotes` sets the characters that must surround the free translation in a gloss. If no value is specified, it will be double quotes. There are no real restrictions for this value.

Second, the following elements can set general styling instructions for different sections of a gloss, formatting them completely in italics OR bold. The items with a | indicate that various names are possible.x

- **source|preamble**: The line of the glosses where the source is rendered.
- **al|first**: The first line of the glosses, with the original language text.
- **bl|second**: The second line of the glosses.
- **cl|third**: The third line of the glosses if it exists.
- **ft|trans|translation**: The line of the glosses where the free translation is rendered.

Each of these items can take one of a few values:

- `i`, `it`, `italics` and `textit` set italics.
- `b`, `bf`, `bold` and `textbf` set boldface.

Third, there are a few LaTeX-specific settings documented in the [expex](#) documentation. In all cases the default value is 0 (Opt). (If you would like other settings to be supported, write up an Issue and I will look into it!)

- **exskip|par_spacing**: Space above *and* below the example. The `par_spacing` name is allowed for backwards compatibility, but the actual name in `expex` is `exskip`.
- **belowglpreambleskip**: Space under the preamble (where the source is printed).
- **aboveglftskip**: The spacing above the free translation.
- **extraglskip**: The spacing between the aligned lines.

Fourth, there is one setting that is not available in LaTeX, particularly thinking of slides: **numbering**, that is, whether the glosses should be numbered (in HTML).

Finally, you may set the following values for Word output, used in computing the spaces for alignment:

- **font_family**: A character vector of length 1 indicating the font family used in the lines to be aligned, or a list with names "a" and "b" (and "c" if relevant) indicating the font families of specific lines.
- **font_size**: A numeric vector of length one indicating the font size used in the lines to be aligned, or a list with names "a" and "b" (and "c" if relevant) indicating the font sizes of specific lines.
- **page_width**: The width of the space between the margins of the Word file, in pixels, used to wrap long examples. The default is 1332; if you see that your output does not match what you want, you can tweak it with this value.

Examples

```
set_style_options(styling = list(a = "b", trans_quotes = ""))
print_config()#'
```

use_glossr	<i>Use glossr</i>
------------	-------------------

Description

Override default configuration.

Usage

```
use_glossr(output_format = NULL, styling = list())
```

Arguments

output_format	Desired output format
styling	Named list of styling options for specific elements of glosses.

Value

Set options

See Also

[set_style_options\(\)](#)

Examples

```
use_glossr(styling = list(numbering = FALSE, trans_quotes = "~"))  
print_config("other")
```

Index

- * **datasets**
 - glosses, [4](#)
- as_gloss, [2](#)
- as_gloss(), [6](#), [10](#)
- config_from_file, [3](#)
- dplyr::filter(), [7](#)
- gloss, [3](#), [4](#), [6](#), [9–11](#)
- gloss_data, [3](#), [5](#), [10](#), [11](#)
- gloss_df, [6](#)
- gloss_df(), [3](#), [6](#), [7](#), [9](#), [10](#)
- gloss_factory, [7](#)
- gloss_factory(), [7](#)
- gloss_format_words, [8](#)
- gloss_html (gloss_render), [10](#)
- gloss_leipzig (gloss_render), [10](#)
- gloss_linetooltip, [8](#)
- gloss_list, [9](#)
- gloss_pdf (gloss_render), [10](#)
- gloss_render, [10](#)
- gloss_render(), [9](#)
- gloss_tooltip (gloss_render), [10](#)
- gloss_word (gloss_render), [10](#)
- glosses, [4](#)
- knit_print.gloss, [11](#)
- knitr::knit_print(), [11](#)
- new_gloss, [11](#)
- new_gloss_data (gloss_data), [5](#)
- print_config, [12](#)
- set_style_options, [12](#)
- set_style_options(), [14](#)
- shiny.tag, [9](#)
- tibble::tibble, [5](#)
- use_glossr, [14](#)
- use_glossr(), [12](#)