Package 'glossr'

June 8, 2022

Type Package

Title Use Interlinear Glosses in R Markdown

Version 0.5.1

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, magrittr, purrr, rlang, stringr,
flextable, tidyr

RoxygenNote 7.2.0

Config/testthat/edition 3

VignetteBuilder knitr

Suggests bookdown, officedown, testthat, rmarkdown, officer, htmltools

Depends R (>= 2.10)

URL `https://montesmariana.github.io/glossr/`,
`https://github.com/montesmariana/glossr`

Language en-GB

NeedsCompilation no

Author Mariana Montes [aut, cre] (<`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 2022-06-08 09:00:05 UTC

R topics documented:

`as_gloss` *Helper to create gloss objects*

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 = getOption("glossr.trans.quotes", "\""),
  output_format = getOption("glossr.output", "latex")
)
```

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.

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 as `data`.

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")
```

`check_packages` *Check if required packages are installed*

Description

Calls `requireNamespace` with the required packages.

Usage

```
check_packages(output_format)
```

Arguments

`output_format`	Word, Leipzig or Tooltip, desired format

`format_html` *Read HTML formatting options*

Description

Read HTML formatting options

Usage

```
format_html()
```

Value

Style tag

`format_pdf` *Read Latex formatting options*

Description

Read Latex formatting options

Usage

```
format_pdf(level)
```

Arguments

`level`	Gloss line to format

Value

Key for expex

`format_word` *Read Word formatting options*

Description

Read Word formatting options

Usage

```
format_word_glosses(ft)

format_word_translation(ft)

format_word_source(source)
```

Arguments

`ft`	`flextable` for gloss lines or translation
`source`	Character vector with the source text.

Value

Formatted table or text

Functions

- `format_word_glosses`: Format glosses
- `format_word_translation`: Format translation
- `format_word_source`: Format source text

`gloss` *Reference gloss*

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` *Examples of glosses*

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` 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_df` *Render gloss from a dataframe*

Description

Render gloss from a dataframe

Usage

```
gloss_df(df, output_format = getOption("glossr.output", "latex"))
```

Arguments

`df`	Dataframe one row per gloss. Columns `translation`, `source` and `label` have special meaning (see `new_gloss_data`); all the others will be interpreted as lines to align in the order given.
`output_format`	(Optional) Whether it will use latex, word or html format.

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_format_words` *Apply latex formatting to many words*

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_linesplit` *Split lines for HTML*

Description

Splits a character string by spaces keeping groups of words surrounded by curly braces together.

Usage

```
gloss_linesplit(line)
```

Arguments

`line`	Character string to split.

Value

Character vector of elements.

`gloss_linetooltip` *Apply tooltip to a full gloss*

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` *Sublist glosses*

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` *Render a gloss*

Description

This 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)

gloss_tooltip(gloss)

gloss_leipzig(gloss)

gloss_word(gloss)
```

Arguments

`gloss`	Object of class `gloss_data`

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)
```

gloss_table *Create table from a gloss*

Description

Create table from a gloss

Usage

```
gloss_table(gloss_output, is_translation = FALSE)
```

Arguments

`gloss_output`	Gloss lines as a table for Word
`is_translation`	Whether the table is for the free translation line.

Value

`flextable` object

`gloss_word_lines` *Process elements of gloss lines for word*

Description

Process elements of gloss lines for word

Usage

```
gloss_word_lines(gloss_lines)
```

Arguments

`gloss_lines`	Unclassed content of a `gloss_data` object

Value

List of tibbles to print

`knit_print.gloss` *Print method for glosses*

Description

Print method for glosses

Usage

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

Arguments

`x`	Object to print
`...`	Other options for knit_print

`latex2html` *Parse latex*

Description

Parse latex

Usage

```
latex2html(string)

latex2word(string, is_cell = TRUE)
```

Arguments

`string`	Latex string to parse
`is_cell`	For word output, whether to style raw text or a cell

Value

Character vector

Functions

- `latex2html`: Convert to HTML
- `latex2word`: Convert to Word

`latex_tag` *Regex for a latex tag*

Description

Regex for a latex tag

Usage

```
latex_tag(tag)
```

Arguments

`tag`	Latex tag

Value

Regex expression to extract tagged string.

`leipzig_script` *Script for leipzig.js*

Description

To append after the first gloss.

Usage

```
leipzig_script()
```

Value

`tag`

`new_gloss` *gloss class*

Description

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

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`

`new_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.

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.

Usage

```
new_gloss_data(
  gloss_lines,
  source = NULL,
  translation = NULL,
  label = NULL,
  trans_quotes = getOption("glossr.trans.quotes", "\"")
)
```

Arguments

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

Value

Object of class `gloss_data`

reset_max *Divide lines for Word*

Description

Helper for `gloss_word`

Usage

```
reset_max(m, c, l)
```

Arguments

`m`	Maximum number of characters for a slot
`c`	Cumulative sum of maximums
`l`	Current line

sc_to_upper *Small caps to upper case*

Description

Replaces small caps tags from LaTeX to upper case within a string.

Usage

```
sc_to_upper(string)
```

Arguments

`string`	String with a LaTeX small caps tag

Value

Character vector of length one

`set_default` *Define a default value*

Description

Define a default value

Usage

```
set_default(x, default = "")
```

Arguments

`x`	Variable to define
`default`	Default value

Value

New value

`set_style_options` *Set general styling options*

Description

This is a helper function to set `options` that control style characteristics for glosses across the full document. It is called within `use_glossr` but can be overridden later but setting the appropriate options.

Usage

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

Arguments

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

Details

There are two 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.

source|preamble The line of the glosses where the `source` is rendered.

a|first The first line of the glosses, with the original language text.

b|second The second line of the glosses.

c|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`, `textit` set italics.
- `b`, `bf`, `bold`, `textbf` set boldface.

#TODO create vignette

Value

Set the appropriate options.

`style_options` *List of styling options*

Description

List of styling options

Usage

```
style_options(format = c("i", "b"))
```

Arguments

`format`	`i` for italics and `b` for bold

Value

Character vector with the ways that a certain format (italics or bold) can be specified.

`tooltip` *Add tooltip to one word*

Description

Add tooltip to one word

Usage

```
tooltip(x, title)
```

Arguments

`x`	Word or expression
`title`	Text of the tooltip

Value

'shiny.tag' with attributes for a tooltip

Examples

```
tooltip("One", "DET.SG")
```

`use_glossr` *Use glossr*

Description

Call in a setup chunk.

Usage

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

Arguments

`html_format`	Whether the html output should use leipzig.js or tooltips.
`styling`	Named list of styling options for specific elements of glosses.

Value

Set options

`use_leipzig` *HTML dependency for leipzig.js*

Description

HTML dependency for leipzig.js

Usage

```
use_leipzig()
```

Value

`htmlDependency`

`use_tooltip` *HTML dependency for tooltip format*

Description

HTML dependency for tooltip format

Usage

```
use_tooltip()
```

Value

`htmlDependency`

`validate_output` *Validate output format*

Description

Validate output format

Usage

```
validate_output(
  output = c("word", "latex", "leipzig", "tooltip", "html", "pdf")
)
```

Arguments

`output`	Character string with output format required.

Value

Invisible, the output. It also sets it as the 'glossr.output' option.

`word_knitr` *Apply officer formatting*

Description

Apply officer formatting

Usage

```
word_knitr(text, bold = FALSE, italic = FALSE)
```

Arguments

`text`	Text to format
`bold`	Whether the word should be in bold
`italic`	Whether the word should be in italics

Value

Knitr-ready text

Index