Using the glossr package for PDF

Mariana Montes

Contents

1 Introduction **1**

2 Basic usage **1**

3 PDF-only features **3**

1 Introduction

The {glossr} package offers useful functions to recreate Leipzig glosses in R Markdown texts. The immediate solution for a *LaTeX* output is to use the {gb4e} library. If that is enough for you, you can still use this package to automatically print them in an R-chunk, avoiding typos, and even generate them automatically from a dataframe with your examples! But chances are, PDF is not enough for you, and you would also like a nice rendering (or at least *some* rendering) of your interlinear glosses in HTML as well. This offers some challenges, because the way to print them is different, the way to reference them is different, and interlinear glosses are a pain to render in HTML.

I took that pain and packaged it so you don't need to feel it.

You can start using `glossr` in a package by calling the library and then `use_glossr()` to activate some background stuff. Mainly, this function informs all the other functions whether you are using *LaTeX* or HTML and which HTML rendering you've chosen. Yes you have a choice! The default one is an implementation of leipzig.js, which is awesome because of the neat alignment, the small caps in morphological annotation and the explanatory tooltips when hovering over them. However, they have the negative consequence that the number of your examples will be next to the translation, rather than on the top line, and you probably can't manipulate the italics on the first line (I haven't tried, actually). Enter the second option, "tooltip", which instead of aligning the morphological notation, it adds it as a tooltip over each of the first line words. This gives you more freedom to adapt the formatting and *might*[1] be more accessible for screen readers.

For the pdf output you might have to install the {gb4e} library with `tinytex::tlmgr_install("gb4e")`.

```
library(glossr) # library(glossr)
use_glossr() # options for html: leipzig (default), tooltip
#> Setting up the `latex` engine.
```

2 Basic usage

When you want to include an example, create a gloss with `as_gloss()` and call it.

[1] I have to test this.

```
my_gloss <- as_gloss(
  original = "Hace calor/frío",
  parsed = "make-3SG-PRS heat/cold-N-A",
  translation = "'It is hot/cold'",
  label = "my-label"
)
my_gloss
```

(1) Hace calor/frío
make-3SG-PRS heat/cold-N-A
'It is hot/cold'

The label given to `as_gloss()` allows you to cross-reference the example: in PDF this is `example (\@ref(my-label))`, whereas in HTML this is `example (@my-label)`. What should YOU do? `gloss("my-label")` can be used inline to generate a reference for either PDF or HTML, depending on the output of your file: (1) in this case.

If you have many examples, you might want to keep them in their own file, if you don't have them like that already. `glossr` offers a small dataset for testing, called `data(glosses)`.

```
library(magrittr)
library(dplyr) # for select() and filter()
data(glosses)
glosses <- glosses %>%
  select(original, parsed, translation, label)
glosses
#> # A tibble: 5 x 4
#>   original                                                parsed translation label
#>   <chr>                                                   <chr>  <chr>       <chr>
#> 1 Mér er heitt/kalt                                       "\\te~ I am hot/c~ feel~
#> 2 Hace calor/frío                                         "make~ It is hot/~ amb-~
#> 3 Ik heb het koud                                         "\\te~ I am cold;~ feel~
#> 4 Kotae-nagara otousan to okaasan wa honobonoto atatak~ "repl~ While repl~ hear~
#> 5 Ainiku sonna shumi wa nai. Tsumetai-none. Kedaru-sou~ "unfo~ Unfortunat~ lang~
```

Assuming you have them in a table with columns matching the arguments of `as_gloss()`, you can give it to `gloss_df()` directly and it will do the job. This table has more columns than we need, so we will only select the right ones and print the glosses of the first three rows. Note that the values in the "label" column will be used as labels: `gloss("feel-icelandic")` will return (2).

```
gloss_df(head(glosses, 3))
```

(2) Mér er heitt/kalt
1SG.DAT COP.1SG.PRS hot/cold.A
I am hot/cold.

(3) Hace calor/frío
make.3SG.PRS heat/cold..N.A
It is hot/cold; literally: it makes heat/cold.

(4) Ik heb het koud
1SG have 3SG COLD.A
I am cold; literally: I have it cold.

3 PDF-only features

This package also offers a few extensions when working on PDF output. On the one hand, `gloss_list()` allows you to nest a list of glosses and have both a reference for the list and for each individual item. **This will not work in HTML**, which will just keep the numbering on the top level. But on PDF, given the function below, we can use `gloss("jp")` to reference (5), or `gloss("heartwarming-jp")` and `gloss("languid-jp")` to reference (5a) and (5b).

```
filter(glosses, endsWith(label, "jp")) %>%
  gloss_df() %>%
  gloss_list(listlabel = "jp")
```

(5) a. Kotae-nagara otousan to okaasan wa honobonoto atatakai2 mono ni tsutsum-areru
reply-while father and mother TOP heartwarming warm thing with surround-PASS
kimochi ga shi-ta.
feeling NOM do-PST

While replying (to your question), Father and Mohter felt like they were surrounded by something heart warming.

b. Ainiku sonna shumi wa nai. Tsumetai-none. Kedaru-souna koe da-tta.
unfortunately such interest TOP not.exist cold-EMPH languid-seem voice COP-PST

Unfortunately I never have such an interest. You are so cold. (Her) voice sounded languid.

Finally, it might be the case that you want to apply *LaTeX* formatting to a long string of elements for your first lines of glosses, e.g. set half of your example in italics. In order to facilitate applying the same formatting to each individual element, this package offers you `gloss_format_words()`, which you can implement to the strings given to `as_gloss()`. Internally, `glossr` will try to parse *LaTeX* formatting into HTML one (because I'm assuming your data is prepared for that). In the future this might be different.

```
gloss_format_words("A long piece of text", "textit")
#> [1] "\\textit{A} \\textit{long} \\textit{piece} \\textit{of} \\textit{text}"
```

```
my_gloss <- as_gloss(
  original = gloss_format_words("Hace calor/frío", "textbf"),
  parsed = "make.3SG.PRS heat/cold.N.A",
  translation = "'It is hot/cold'",
  label = "formatted"
)
my_gloss
```

(6) **Hace** **calor/frío**
make.3SG.PRS heat/cold.N.A

'It is hot/cold'