Working with bibliography in texor

by Abhishek Ulayil

Abstract This is a small sample article to demonstrate usage of **rebib** to manage bibliographies.

1 Introduction

The **rebib** package is a spun-off from **texor** package.

The decision to do this is the fact that the bibliography section in **texor** package was expanding significantly, enough to deserve a dedicated space.

Thus **rebib** was born and has improved over the past few months to include more advanced features and utilities.

What does rebib do?

In LaTeX, we have multiple ways to declare bibliography entries, which include embedding bibliography entries in the document itself or isolating that section to an external file commonly called a bbl file. Then we have the BibTeX format, which has its structure and is much superior in managing bibliographic entries.

LaTeX has no problem dealing with multiple formats and works well with all these bibliographic formats. However, some web publishing journals like RJ web articles from RJournal do not support embedded bibliographies. As a result, they need a BibTeX equivalent of the same.

The **rebib** package converts embedded LaTeX bibliographies into a close BibTeX equivalent. This package eases the author from the manual conversion of the documents and aids the texor project in automating the conversion of past RJournal articles to the RJ-web-article format.

features of rebib

- Reads bib chunks to produce a very close BibTeX equivalent
- Title and author are usually mandatory fields
- URL, ISBN, publisher, pages and year are optional fields and will be enabled when relevant
- Rest of the data is stored in `"journal"`(internally) and `"publisher"`(when writing BibTeX file)
- Ignores commented LaTeX code
- Citation tracker
- Bibliography Aggregation

2 Parser

The magic behind **rebib** is a parser which parses bibliographic data and assorts them according to the matching regular expressions.

Stage 1: Read the embedded bibliography block

This stage is a minor step where it reads the Embedded Bibliography from the LaTeX document. This step also includes Filtering out the commented code to avoid un-intended entries read.

Lastly, the data is broken down based on the LaTeX macro `\\bibitem` as a marker for a new entry and this assorted data is exported to a variable.

Setup :

```
dir.create(your_article_folder <- file.path(tempdir(), "exampledir"))
example_files <-  system.file("article", package = "rebib")
x <- file.copy(from = example_files,to=your_article_folder,recursive = T)
your_article_path <- paste(your_article_folder,"article",sep="/")
```

Input :

```
file_name <- rebib:::get_texfile_name(your_article_path)
bib_items <- rebib:::extract_embeded_bib_items(your_article_path,file_name)
bib_items[[1]]
```

Output :

```
#> [1] "\\bibitem[Ihaka, Ross and Gentleman, Robert]{ihaka:1996}"
#> [2] "Ihaka, Ross and Gentleman, Robert"
#> [3] "\\newblock \\emph{R: A Language for Data Analysis and Graphics.}"
#> [4] "\\newblock \\emph{Journal of Computational and Graphical Statistics}, 3:\\penalty0"
#> [5] "299--314, 1996."
#> [6] "\\newblock URL : \\url{https://doi.org/10.1080/10618600.1996.10474713}"
```

Stage 2: Regex powered parser

Now, with the chunks of bibliographic entries, each is passed to a parser which will break it down based on regular expressions. The logic is to use the LaTeX macro `\\newblock` as a placeholder to identify the position of text elements relative to it.

The first value to be parsed is the `unique_id` also called the citation reference which is used to cite elements inside the article. Usually, this is in the first or second line of the whole entry. The position of the `unique_id` will determine the position of the author names.

Input :

```
bib_items[[1]][1]
```

Output :

```
#> [1] "\\bibitem[Ihaka, Ross and Gentleman, Robert]{ihaka:1996}"
```

After reading the `unique_id`, the parser will attempt to read the author name(s).

Input :

```
bib_items[[1]][2]
```

Output :

```
#> [1] "Ihaka, Ross and Gentleman, Robert"
```

Next, the title is extracted based on the position of the new blocks or the end of the bib chunk.

Input :

```
bib_items[[1]][3]
```

Output :

```
#> [1] "\\newblock \\emph{R: A Language for Data Analysis and Graphics.}"
```

This way the crucial elements of the bibliographic entry (unique$_i d, authornamesandtitle)areparsedout.Theremainingdataisstor$

Input :

```
bib_items[[1]][4:6]
```

Output :

```
#> [1] "\\newblock \\emph{Journal of Computational and Graphical Statistics}, 3:\\penalty0"
#> [2] "299--314, 1996."
#> [3] "\\newblock URL : \\url{https://doi.org/10.1080/10618600.1996.10474713}"
```

A series of filters for ISBN, URL, pages and year fields are applied to search for relevant data from the remaining data. If the data is not available then it is set as NULL and will be skipped while writing the BibTeX file. There is a lot of filtering of common LaTeX elements and after that, the data remaining is stored in a structured format to be written to a file.

Input :

```
bib_entry <- rebib:::bib_handler(bib_items)
bib_entry
```

Output :

```
#> $book
#> $book[[1]]
#> $book[[1]]$unique_id
#> [1] "ihaka:1996"
#>
#> $book[[1]]$author
#> [1] "Ihaka, Ross and Gentleman, Robert"
#>
#> $book[[1]]$title
#> [1] "R: A Language for Data Analysis and Graphics"
#>
#> $book[[1]]$journal
#> [1] "Journal of Computational and Graphical Statistics 3:   :"
#>
#> $book[[1]]$year
#> [1] "1996"
#>
#> $book[[1]]$URL
#> [1] "https://doi.org/10.1080/10618600.1996.10474713"
#>
#> $book[[1]]$isbn
#> NULL
#>
#> $book[[1]]$pages
#> [1] "299--314"
#>
#>
#> $book[[2]]
#> $book[[2]]$unique_id
#> [1] "R"
#>
#> $book[[2]]$author
#> [1] "R Core Team"
#>
#> $book[[2]]$title
#> [1] "R: A Language and Environment for Statistical Computing"
#>
#> $book[[2]]$journal
#> [1] "R Foundation for Statistical Computing Vienna Austria    :"
#>
#> $book[[2]]$year
#> [1] "2016"
#>
#> $book[[2]]$URL
#> [1] "https://www.R-project.org/"
#>
#> $book[[2]]$isbn
#> [1] "3-900051-07-0"
#>
#> $book[[2]]$pages
#> NULL
```

Stage 3: BibTeX writer

After reading the bibliographic entries and splitting out meaningful values from them, we can finally write a structured file in the BibTeX format.

The writer will read the bib chunks one at a time based on the metadata extracted and will write the corresponding data fields. The default entry type is a book, which should not have any problems with the web articles.

Input :

```
file_path <- paste0(your_article_path,"/",file_name)
bib_path <- paste0(your_article_path,"/example.bib")
```

Input :

```
rebib:::bibtex_writer(bib_entry,file_path)
cat(readLines(paste(your_article_path,"example.bib",sep="/")),sep = "\n")
```

Output :

```
#> @book{ihaka:1996,
#> author = {{Ihaka, Ross and Gentleman, Robert}},
#> title = {{R: A Language for Data Analysis and Graphics}},
#> publisher = {Journal of Computational and Graphical Statistics 3:   :},
#> pages = {299--314},
#> year = {1996},
#> url = {https://doi.org/10.1080/10618600.1996.10474713}
#> }
#> @book{R,
#> author = {R {Core Team}},
#> title = {{R: A Language and Environment for Statistical Computing}},
#> publisher = {R Foundation for Statistical Computing Vienna Austria    :},
#> year = {2016},
#> url = {https://www.R-project.org/},
#> isbn = {3-900051-07-0}
#> }
```

I expect the authors who are converting the document to take a look and check if there are any errors or missing values.

3 General usage

4 Bibliography aggregation

5 Summary

In summary the **rebib** package supports:

- Conversion of embedded natbib type bibliography used in RJournal
- Aggregation of bibliographic entries from the article as well as BibTeX

Abhishek Ulayil
Student, Institute of Actuaries of India
Mumbai, India
ORCiD: 0009-0000-6935-8690