Converting LaTeX Legacy R Journal Articles into R Markdown Articles using texor and rebib

by Abhishek Ulayil, Mitch O'Hara-Wild, Christophe Dervieux, Heather Turner, and Dianne Cook

Abstract In 2021 the R Journal made a change of templates for article writing to R Markdown instead of LaTeX. The reasons were to encourage better reproducibility of articles using dynamic documents, enable interactivity in articles, and to make the articles more accessible for vision-impaired readers. A resulting challenge was to explore whether legacy articles might be suitably converted into R Markdown before converting into HTML output. This paper describes the process to convert an R Journal LaTeX template article into an R Markdown article, and the two new R packages, texor and rebib, that can be used to achieve the conversion.

1 Introduction

The R Journal is the primary open-access outlet for publications produced by the R community. It was born in 2008, evolving from a newsletter, that ran from 2001, into a more formal article publication to encourage documenting statistical computing research.

The format is regularly updated. Early articles were typeset using LaTeX (The LaTeX Project 2023), from a specific, but changing, template. This requires that code is separated from the documentation, and there is a chance that code chunks in the paper don't reproduce the results reported. With the emergence of dynamic document systems like R Markdown (Xie, Allaire, and Grolemund 2018) a tight-coupling of code and documentation is possible. Code chunks are dynamically executed when the document is typeset using a system like knitr (Xie 2015), making reporting of computing research more reproducible.

In 2019, with the help of funding from the R Consortium it was decided that it was time to update operations. One aspect of this was to change from LaTeX paper submissions to a more reproducible format, where code was embedded in the document, and the output could be both HTML and pdf. There are numerous benefits of HTML format:

1. Articles can include interactive graphics and tables.
2. The format is more accessible to screen readers making the work more accessible to vision-impaired researchers.

This latter point is a reason to consider converting all of the legacy articles into HTML.

A key decision for creating conversion software was to decide to directly convert LaTeX to HTML, or PDF to HTML, or LaTeX to R Markdown, and then use the current journal tools to create the HTML. The latter approach was decided to be the most versatile and useful. If an article can be converted from LaTeX to R Markdown, it would help authors make the transition to reproducible publishing, beyond what the R Journal needed. Once an article is in R Markdown format it can be adapted to include the code for dynamic execution.

In addition to article format, changes to the web site structure were important for delivering the publication. Web site architectures are also constantly evolving, and the emergence of distill (Dervieux et al. 2022) allows for the journal web site to optimally deliver R Markdown articles.

The rjtools was developed to create articles using R Markdown for the R Journal, and to embed them into the journal web site. The packages described here, texor and rebib describes software to convert legacy LaTeX format articles into Rmarkdown, so that they can be rendered in HTML in the new web site.

The paper is organised as follows. Section 2 gives an overview of the conversion process, that includes pre-processing using regular expressions, post-processing using Lua filters, and handling of figures, tables and equations. Section 3 describes the texor that handles most of the conversion. Section 4 describes tools to do special handling of bibliography files. The supplementary materials have folders containing specific examples that can be used for understanding how the conversions are done.
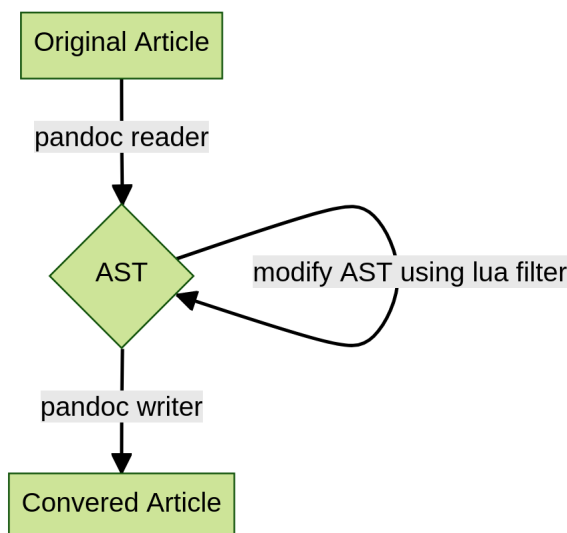
```
        ┌──────────────────┐
        │ Original Article │
        └──────────────────┘
                 │
           pandoc reader
                 │
                 ▼
              ◆ AST ◆  ⟲ modify AST using lua filter
                 │
           pandoc writer
                 │
                 ▼
        ┌──────────────────┐
        │ Convered Article │
        └──────────────────┘
```

Figure 1: Workflow of the document conversion conducted by texor.

2   The internals of converting from LaTeX to R Markdown

The decision to convert to R Markdown format means that the final output to pdf and HTML will depend on Pandoc (MacFarlane 2023). Pandoc is a versatile document conversion program written in Haskell that is core to numerous documentation systems, including R Markdown and Quarto. Pandoc first converts a document into an abstract syntax tree. From this, it can convert to a different format, including custom ones.

Pandoc can be used to do the conversion from LaTeX to R Markdown also. However, additional pre-processing needs to be done to handle special R Journal LaTeX styling. And further post-processing needs to be done to handle specific R Journal R Markdown styling. The **texor** package contains functionality to handle this pre- and post-processing of the document, in a workflow illustrated in Figure 1.

Pre-processing using regular expressions

LaTeX is very descriptive language, that allows authors substantial freedom for customization. Markdown (Gruber 2002), on which R Markdown is based, is more restrictive and was born to make it easier to create web pages without the distraction of a gazillion HTML tags. The beauty of Markdown is that it allows the author to focus on writing, without format cluttering the text. The drawback is that it is simple typesetting, optimized for web delivery.

While Pandoc can do most of the heavy-lifting, it cannot cope with all the freedom with which LaTeX documents are written. An example of this is with formatting of code. Pandoc only handles the verbatim environment, but there are many ways to format code in LaTeX, and the R Journal template has a special \code{} command. If the code environment is not verbatim, then Pandoc will also try to process the actual code content as LaTeX commands and will likely lose details. It is better to convert these synonyms into verbatim environments this prior to passing the document to Pandoc.

The functions in **texor** that handle the pre-processing using regular expressions are:

- stream_editor(): operates like the sed function in unix (Ritchie and Thompson 1978) and allows generic text pattern matching and replacing.
- patch_code_env(): replaces the common code environments, code, example, Sin, Sout, Scode, Sinput, smallverbatim, boxedverbatim, smallexample with verbatim.
- patch_equations(): coordinates various equation environments.
- patch_figure_env(): coordinates various figure environments.
- patch_table_env(): coordinates table environments.

These functions are verbose and describe all the changes being made. They also create a backup of the original file before making the changes.

Post-processing using Lua filters

Lua (Ierusalimschy, Figueiredo, and Filho 1996) is a programming language, that is light-weight, fast, ideal for procedural operations. It is embedded in many other applications to allow custom scripting for extensibility. Pandoc allows users to provide custom Lua filters to produce custom output formats. The **texor** package handles post-processing of the R Markdown document into the special format for the R Journal using a suite of Lua filters.

Here is an example of a Lua filter available in **texor**:

```
function Div(el)
    if el.classes[1] == 'thebibliography' then
        return { }
    end
end
```

This filter reads the abstract syntax tree, selecting all the Div elements. Then it looks for the class "thebibliography." This Div element contains the LaTeX bibliographic records, that appear at the very end of papers. It should not be in the document when using the "RJ-web-article" layout, because it is added from meta-data when the R Markdown is knitted. So the Lua filter removes this section.

Figures

Standard, single figure

Figure definitions in LaTeX are many and varied! The standard, single figure definition with the figure environment and raster image format such as PNG or JPG, is handled by Pandoc. It will convert:

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.35\textwidth]{Rlogo-5.png}
  \caption{The logo of R.}
  \label{figure:rlogo}
\end{figure}
```

to

```
<figure id="figure:rlogo">
<img src="Rlogo-5.png" style="width:35.0%" />
<figcaption>Figure 1: The logo of R.</figcaption>
</figure>
```

PDF format images

Images in PDF format are converted to PNG, in the pre-processing of the LaTeX document, and then post-processed using Pandoc as described above.

Multiple figures

Multiple figures are supported with the latest versions of Pandoc, so definitions like:

```
\begin{figure*}[htbp]
  \centering
  \includegraphics[width=0.45\textwidth]{Rlogo-5.png}
  \includegraphics[width=0.45\textwidth]{normal}
  \caption{Images side by side}
  \label{fig:twoimages}
\end{figure*}
```

will be converted to:

```
<figure id="fig:twoimages">
<p><img src="Rlogo-5.png" style="width:45.0%" alt="image" /><img
src="normal.png" style="width:45.0%" alt="image" /></p>
<figcaption>Figure 3: Images side by side</figcaption>
</figure>
```

tikz format images

Some legacy articles define images using tikz commands, such as:

```
\begin{figure}

%% Generated Image will included as a PNG above automatically
  \centering
\tikzstyle{process} = [rectangle, rounded corners,
minimum width=3cm,
minimum height=1cm,
text centered,
draw=black]
\tikzstyle{arrow} = [thick,->,>=stealth]
\begin{tikzpicture}[node distance=4cm]
%Nodes
...
```

This is handled by pre-processing the LaTeX to create the image, as both PDF, and then PNG, for inclusion in the R Markdown document using:

```
<figure id="fig:tikz">
<img src="tikz/figtikz.png" style="width:100.0%" />

<figcaption>Figure 5: Tikz Image example</figcaption>
</figure>
```

Algorithms as figures

Algorithms as figures is supported, and the following description will yield the result in Figure 2.

```
\begin{algorithm}[htbp]
\SetAlgoLined
\KwData{this text}
\KwResult{how to write algorithm with \LaTeX2e }
initialization\;
\While{not at end of this document}{
read current\;
\eIf{understand}{
go to next section\;
current section becomes this one\;
}{
go back to the beginning of current section\;
}
}
\caption{How to write algorithms}
  \label{alg:how}
\end{algorithm}

<figure id="alg:how">
<img src="alghow.png" style="width:100.0%" />
<p>initialization</p>
<figcaption>Algorithm 1: How to write algorithms</figcaption>
</figure>
```

> **Data:** this text
> **Result:** how to write algorithm with LaTeX2e
> initialization;
> **while** *not at end of this document* **do**
> > read current;
> > **if** *understand* **then**
> > > go to next section;
> > > current section becomes this one;
> >
> > **else**
> > > go back to the beginning of current section;
> >
> > **end**
>
> **end**

Figure 2: How to write algorithms.

Equations

Math is handled primarily by Pandoc. The inline math and equation descriptions are unchanged between LaTeX and R Markdown.

The HTML output renders math using MathJax. This does mean that some functionality, like `\bm`, `\boldmath` and `mathbb` not supported, and special definitions cannot be handled.

The numbering of equations is a bit trickier. LaTeX automatically numbers equations, unless specifically instructed not to. Equation numbering in R Markdown requires specific labeling using (\#eq:xx) as described in Yihui Xie (2023). The **texor** helps by adding the labeling using a Lua filter to convert the existing `\label{..}` to (\#eq:xx).

Tables

Tables form one of the biggest challenges in migrating from LaTeX to R Markdown, because the sophistication is not completely replicated. However, there have been many improvements in table definitions for R Markdown that are increasing producing the beautifully crafted tables possible in LaTeX. The conversion in **texor** can mostly handle the simple tables, and for producing more complex tables it may be necessary to manually edit the resulting `Rmd` file to make conditional tables, one to render specifically for HTML output using packages such as **kableExtra** (Zhu 2021), **gt** (Iannone et al. 2023), **htmlTable** (Gordon, Gragg, and Konings 2022), **tableHTML** (Boutaris, Zauchner, and Jomar 2023), **tables** (Murdoch 2023) or **DT** (Xie, Cheng, and Tan 2023). A benefit of special HTML table output is interactivity, that would allow the table to be sorted in different ways.

Generic tables

Simple LaTeX tables are converted into traditional markdown format tables, by Pandoc. So this table definition:

```
\begin{table}[htbp]
\centering
\begin{tabular}{l | llll }
 \hline
 Graphics Format & LaTeX & Markdown & Rmarkdown & HTML \\
 \hline
 PNG       & Yes & Yes & Yes & Yes \\
 JPG       & Yes & Yes & Yes & Yes \\
 PDF       & Yes & No & No & No \\
 SVG       & No & Yes & Yes & Yes \\
 Tikz      & Yes & No & Yes & No \\
 Algorithm & Yes & No & No & No \\
\hline
\end{tabular}
\caption{Image Format support in various Markup/Typesetting Languages}
\label{table:1}
\end{table}
```

will be converted to:

```
::: {#table:1}
  -------------------------------------------------
  Graphics Format   LaTeX   Markdown   Rmarkdown   HTML
  ----------------- ------- ---------- ----------- ------
  PNG               Yes     Yes        Yes         Yes

  JPG               Yes     Yes        Yes         Yes

  PDF               Yes     No         No          No

  SVG               No      Yes        Yes         Yes

  Tikz              Yes     No         Yes         No

  Algorithm         Yes     No         No          No
  -------------------------------------------------

  : Table 1: Image Format support in various Markup/Typesetting
  Languages
:::
```

Multicolumn tables

A multicolumn table requires:

1. The stream editor modifies the \multicolumnx{..} to \multicolumnxx{..}.
2. A LaTeX macro is used to redefine the \multicolumnxx{..} to \multicolumnx{---}.
3. Pandoc reads the table and transforms it to markdown.

EXAMPLE	X		Y	
	1	2	1	2
EX1	X11	X12	Y11	Y12
EX2	X21	X22	Y21	Y22
EX3	X31	X32	Y31	Y32
EX4	X41	X42	Y41	Y42
EX5	X51	X52	Y51	Y52

Table 1: An example multicolumn table.

Also note that the stream editor is used to rename `table*` environment to `table` environment because the markdown/HTML will not support the specific changes table* environment will bring.

Other tables

Tables with images, math, code or links in the cells are generally handled. Also `widetable` tables that allow for specific width or wrapping of tables into blocks are also partially handled.

Animation

XXX Shall we describe the one pdf paper that has animation and how this is handled - could be cool to include

Logging the conversion

XXX what should go here?

3   Using texor

The package texor can be installed from CRAN using:

```
install.packages("texor")
```

and the development version from https://github.com/Abhi-1U/texor. The website for the package, https://abhi-1u.github.io/texor, has vignettes documenting usage.

Note that you will need to use Pandoc Version > 3.0.0 (if possible latest) for the best results. You can check your version with:

```
rmarkdown::pandoc_version()
```

The only function that a user necessarily needs is latex_to_web(). This creates the R Journal style R Markdown file from a given R Journal style LaTeX file. This is achieved by several sequential steps, convert_to_markdown(), generate_rmd() and produce_html().

The supplementary materials have several folders with specific examples of figures, tables, equations that can be used to understand how the conversion operates.

For converting the 14 years of legacy R Journal articles, batch processing of issues was conducted. This can be achieved using:

XXX

For individuals who are interested in submitting their paper to the R Journal but have written their article using the legacy LaTeX format and wish to convert it to the new format, you can use the latex_to_web() function on your paper directory.

4   Managing the bibliography using rebib

Typically bibliographies are generated during the processing of a LaTeX article using the BibTeX software (Feder 2006) operating on a .bib list of references. The R Journal template requires the inclusion of the .bib file. But LaTeX actually uses a .bbl format for references, which is what BibTeX generates as an intermediate format during the article processing.

During the conversion of legacy articles, especially in the earliest published articles, it was discovered that some papers were only accompanied by the intermediate .bbl formatted references and often these were included directly in the .tex file. Sometimes articles had both references in the .tex and separate .bbl or .bib files. While LaTeX can technically handle either .bbl or .bib formatted references R Markdown can only handle .bib.

The rebib package was developed to handle these tricky situations. It converts embedded LaTeX bibliographies into a close BibTeX equivalent. The features of the package are:

- reading bbl 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 available.
- remaining data is stored in "journal" (internally) and "publisher" (when writing BibTeX file).
- ignores commented LaTeX code.
- provides a citation tracker
- does bibliography aggregation XXX what is this?

The package rebib can be installed from CRAN using:

```
install.packages("rebib")
```

and the development version from https://github.com/Abhi-1U/rebib. The website for the package, https://abhi-1u.github.io/rebib, has vignettes documenting usage.

5   Summary

Remember only the HTML is rendered, and the original PDF is not re-generated.

Only handling to HTML format, especially relevant for figures, because the legacy pdf will remain the same. If using for converting work from LaTeX to Rmd then this will need manual changing by author. See rjtools guide.

Where this might be useful in the future, other applications.

No shift to quarto yet.

Known problems that need manual fixes.

Acknowledgments

Supplementary materials

The supplementary materials has example folders containing LaTeX documents that allow the reader to see how different common patterns in the legacy documents are handled with the conversion. These include:

- `code-env`: Explains how different code environments defined by the R Journal style are handled, and additional details such as code in figure environments, and code in table environments.
- `math-env`: Examples of inline math, display math, how equation numbering is handled by a Lua filter to convert from LaTeX labeling to R Markdown labeling.
- `figure-env`: Explains how the variety of figure definitions are handled in the conversion, including different image formats, numbering, captions, labeling, multiple images, and `tikz` (Josh Cassidy 2013) images.
- `table-env`: Examples of how a variety of table types are converted, including multicolumn, complex and wide tables.
- `lua-filters`: Overview and lots of small examples of Lua filters to handle the custom output needed for the R Markdown format.
- `metadata`: This has a collection of additional format handling including extracting metadata like author names and affiliations, article identifiers used in the review process, and handling citations, footnotes and links.
- `bibliography`: The bibliography was handled differently over the years of the journal, and this details how to use the `rebib` functionality to handle `bbl` files, embedded `bbl`, to convert into the standard `.bib` format.

In each of these folders there is a `RJwrapper.tex`, and `.tex` file, with the extra template files `RJournal.sty` and `Rlogo-5.png` and `.bib` files. These match the legacy template file structure, from which the `RJwrapper.pdf` file is created. To test the conversion for each of these examples, set the path directory to one of the folders and use the `latex_to_web()` function as follows:

```
article_dir <- "path-to-this supplementary folder"
texor::latex_to_web(article_dir)
```

This will create an `.Rmd` and `.html` files in the same directory, that demonstrate the converted R Markdown version and the HTML output format.

You'll need to ensure that you have the latest versions of **texor** and **rebib**, and Pandoc (at least later than version 3.0.0).

Source materials

The **texor** and **rebib** source code and materials to reproduce this paper are available at:

- **texor**: https://abhi-1u.github.io/texor
- **rebib**: https://abhi-1u.github.io/rebib/
- This paper: https://github.com/Abhi-1U/texor-rjarticle
- More details on **rjtools** are at https://rjournal.github.io/rjtools/

References

Boutaris, Theo, Clemens Zauchner, and Dana Jomar. 2023. *tableHTML: A Tool to Create HTML Tables*. https://CRAN.R-project.org/package=tableHTML.

Dervieux, Christophe, JJ Allaire, Rich Iannone, Alison Presmanes Hill, and Yihui Xie. 2022. *Distill: 'R Markdown' Format for Scientific and Technical Writing*. https://CRAN.R-project.org/package=distill.

Feder, Alexander. 2006. "BibTeX: Reference Management Software." https://www.bibtex.org.

Gordon, Max, Stephen Gragg, and Peter Konings. 2022. *htmlTable: Advanced Tables for Markdown/HTML*. https://CRAN.R-project.org/package=htmlTable.

Gruber, John. 2002. "Markdown." https://daringfireball.net/projects/markdown/.

Iannone, Richard, Joe Cheng, Barret Schloerke, Ellis Hughes, Alexandra Lauer, and JooYoung Seo. 2023. *Gt: Easily Create Presentation-Ready Display Tables*. https://CRAN.R-project.org/package=gt.

Ierusalimschy, Roberto, Luiz Henrique de Figueiredo, and Waldemar Celes Filho. 1996. "Lua—an Extensible Extension Language." *Software: Practice and Experience* 26 (6): 635–52.

Josh Cassidy. 2013. *LaTeX Graphics using TikZ: A Tutorial for Beginners (Part 3)—Creating Flowcharts*. Overleaf tutorials. https://www.overleaf.com/learn/latex/.

MacFarlane, John. 2023. "Pandoc: A Universal Document Converter." https://pandoc.org.

Murdoch, Duncan. 2023. *Tables: Formula-Driven Table Generation*. https://CRAN.R-project.org/package=tables.

Ritchie, O. M., and K. Thompson. 1978. "The UNIX Time-Sharing System." *The Bell System Technical Journal* 57 (6): 1905–29. https://doi.org/10.1002/j.1538-7305.1978.tb02136.x.

The LaTeX Project. 2023. "LaTeX – a Document Preparation System." https://www.latex-project.org.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Joe Cheng, and Xianying Tan. 2023. *DT: A Wrapper of the JavaScript Library 'DataTables'*. https://CRAN.R-project.org/package=DT.

Yihui Xie. 2023. *bookdown: Authoring Books and Technical Documents with R Markdown*. #equations. https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html.

Zhu, Hao. 2021. *kableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.

Abhishek Ulayil
Institute of Actuaries of India (student)
Mumbai, India
ORCiD: 0009-0000-6935-8690
perricoq@outlook.com

Mitch O'Hara-Wild
Monash University
Melbourne, Australia
ORCiD: 0000-0001-6729-7695
mail@mitchelloharawild.com

Christophe Dervieux
Posit PBC
Paris, France
ORCiD: 0000-0003-4474-2498
christophe.dervieux@gmail.com

Heather Turner
University of Warwick
Newport, United Kingdom
ORCiD: 0000-0002-1256-3375
ht@heatherturner.net

Dianne Cook
Monash University
Melbourne, Australia
ORCiD: 0000-0002-3813-7155
dicook@monash.edu