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 HTML output. This paper describes the process to convert an R Journal article from LaTeX to HTML format via R Markdown, 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 started in 2009, evolving from the R News newsletter that ran from 2001, to become a more formal publication that encourages and provides credit for the documentation of statistical computing research.

The tooling behind the production of the R Journal is regularly updated. Early articles were typeset using LaTeX (The LaTeX Project 2023), based on a specific, but changing, template. Using LaTeX 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 such as 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, work began to update operations. One aspect of this was to change from LaTeX article 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 and other assistive technologies, making the work more accessible to vision-impaired researchers.
3. HTML provides a more comfortable reading experience on mobile devices, which are increasing used as researchers work on the move and share work via social media.
4. Search engines can easily access the full text of articles, facilitating discovery of published articles.

The latter points are motivations for converting all of the legacy R Journal articles into HTML.

A key decision when designing the conversion software, was whether to convert the LaTeX source to HTML; the PDF output to HTML; or the LaTeX source to R Markdown, which would then be converted to HTML using the current journal tools. 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 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**, describe software to convert legacy LaTeX format articles into R Markdown, 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** package that handles most of the conversion. Section 4 describes tools to do special handling of bibliography files. The supplementary materials

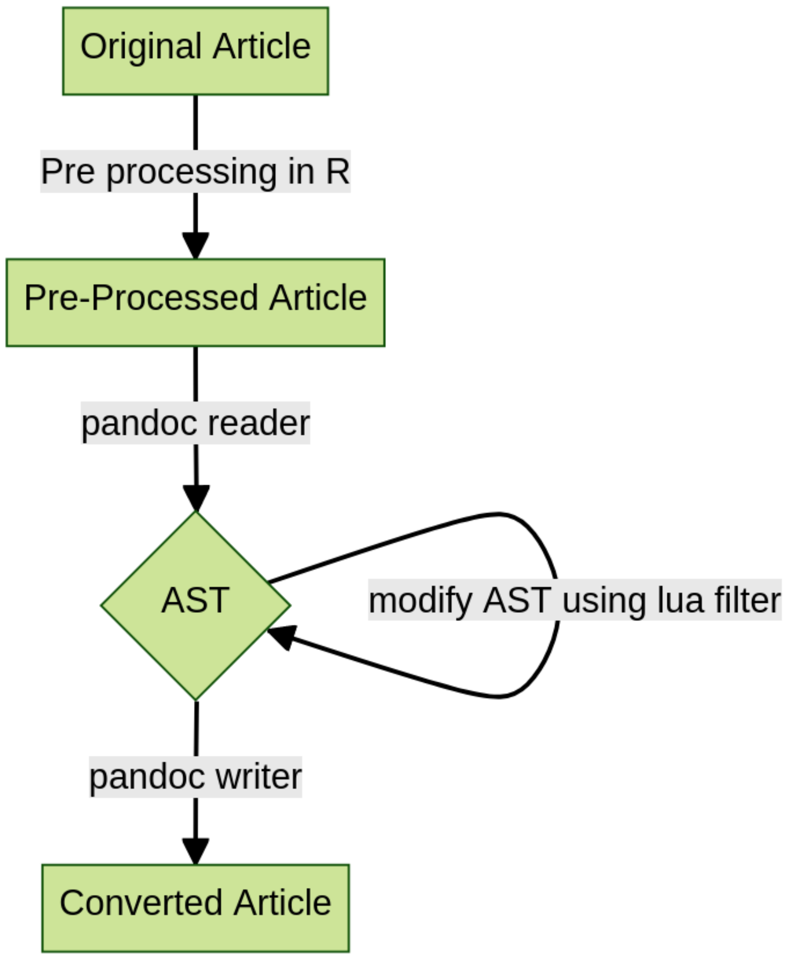

Figure 1: Workflow of the document conversion conducted by texor.

have folders containing specific examples that can be used for understanding how the conversions are done.

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, Krewinkel, and Rosenthal 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, some pre-processing is necessary to handle special R Journal LaTeX styling. And further post-processing is necessary 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 designed 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 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 replacement.
- patch_code_env(): replaces the common code environments, code, example, Sin, Sout, Scode, Sinput, smallverbatim, boxedverbatim, smallexample with verbatim.
- patch_equations(): handles various equation environments.
- patch_figure_env(): handles various figure environments.
- patch_table_env(): handles various 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 and 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 are 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\;
}{
```

```
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.

```
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>
```

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 `\mathbbm` are not supported, and special definitions can only be handled in a limited capacity.

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 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 using conditional markup is that it can take advantage of HTML-specific features, such as sortable columns or paged tables.

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 `\multicolumn{..}` to `\multicolumnx{..}`.
2. A LaTeX macro is used to redefine the `\multicolumnxx{..}` to `\multicolumn{---}` (which is accepted by pandoc).
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 HTML format is single column, so the asterisk indicating that the table should be drawn over the full width of the page is redundant in this case.

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.

3 Using texor

The package `texor` can be installed from CRAN 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 will typically require 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()`.

For converting the 14 years of legacy R Journal articles, batch processing of issues was conducted.

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 current R Markdown format, you can use the `latex_to_web()` function on your paper directory. This will get you about 80% of the way to an R Markdown version of your paper. You will then want to

- Edit the lines where figures are included. The conversion will create HTML code to define the image. This should be changed into Markdown description `![Caption](Image)` or using `knitr::include_graphic(Image)` in an R code chunk, in order for both HTML and PDF versions of the paper to be created when the `rjtools` article template is knitted.
- Edit the tables for aesthetics. The conversion will create Markdown tables, which will convert appropriately to HTML and PDF. However, to have more control and to create more elegant tables using `knitr::kable` provides finer level control.
- Include your R code, to dynamically do the computing described in your article. If any code block is time-consuming to complete, saving intermediate output, or caching would be recommended.

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 current 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.

The instructions for R Journal authors have changed over time. Initially, authors were instructed to run BibTeX on their article and to include the `.bbl` formatted references directly in the `.tex` file. This was to avoid clashes in citation keys between multiple articles in an issue. Later on, authors were instructed to provide a `.bib` file instead. During the conversion of legacy articles, it was discovered that some papers had references in both the `.tex` file and in separate `.bbl` or `.bib` files. Usually, the different files were equivalent, but sometimes references were only found in one source and in some cases the files contained conflicting information! 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:

- extracting embedded bibliographic entries from a `.tex` file;
- creating the mandatory title and author fields;
- creating the optional URL, ISBN, publisher, pages and year fields, when available;
- storing remaining information in `"journal"` (internally) and `"publisher"` (when writing BibTeX file);
- ignoring commented LaTeX code;
- tracking citations included in the document, and
- aggregating references from embedded bibliographic entries with references from supplementary `.bbl` and `.bib` files.

The package rebib can be installed from CRAN 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 The process of converting all the legacy articles

6 Known problems needing manual fixes

7 Summary

The original motivation for the **texor** and **rebib** packages was to convert legacy LaTeX articles into HTML format to accessibility. Thus it only creates the R Markdown and an HTML version leaving the original PDF as published. This will also be used to produce HTML versions of newly accepted papers only submitted as LaTeX.

However, it could be useful to help authors convert to dynamic documents, by providing an initial R Markdown version of their LaTeX, that with some modification, such as including their R code for computations directly in the document, will produce both PDF nd HTML versions of their submitted paper.

Since the shift to R Markdown, a new development for the open source community has emerged: quarto (Allaire et al. 2022). Although R Markdown would suggest a focus on R, it was always possible to include code chunks written in other languages. But Quarto makes this cleaner, and thus more appealing for non-R developers. It also provides a cleaner type-setting. At some point, the R Journal will likely shift to a Quarto template, which is reasonably straightforward, but for the present R Markdown is a suitable dynamic document delivery system for the R Journal.

Acknowledgments

The authors wish to thank the Google Summer of Code program for financially supporting Abhishek Ulayil's work on this project, and the R Project organization for their support. Heather Turner was supported by the Engineering and Physical Sciences Research Council [EPSRC EP/V052128/1] during this project.

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 (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**: Version 1.2.0 `https://abhi-1u.github.io/texor`
- **rebib**: Version 0.3.0 `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

Allaire, J. J., Charles Teague, Yihui Xie, and Christophe Dervieux. 2022. "Quarto." Zenodo. `https://doi.org/10.5281/zenodo.5960048`.

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

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

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.

MacFarlane, John, Albert Krewinkel, and Jesse Rosenthal. 2023. "Pandoc." https://github.com/jgm/pandoc.

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/`.

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

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`.

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
Coventry, United Kingdom
ORCiD: 0000-0002-1256-3375
ht@heatherturner.net

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