Data Publication 1

Ryan Womack

2025-10-21

Table of contents

creating reproducible research and literate programming documents, using knitr, Sweave, Quarto, blogdown, and bookdown

Note that the Data Publication 2 workshop covers data sharing and the creation of data packages.

1 Why Data Publication?

This workshop discusses the initial creation of data publications that embody the best practices of reproducible research and literate programming. In the context of the R statistical programming environment, the knitr, Sweave, blogdown, and bookdown packages are illustrated.

In brief, we want to ensure that the data behind a publication gets the same careful attention, visibility, and usefulness, as the written article (or other form of publication) that uses the data. This is necessary so that others can understand, validate, and possibly reuse the data in future research.

2 Setup

To run the R packages referred to below, install them if you have not already done so.

```r
library(pak)
pkg_install("blogdown")
pkg_install("bookdown")
```

3 Reproducibility

3.1 Credibility in Science

A foundational principle of the scientific method is that when we repeat or redo experiments and scientific analysis, we have the same findings, proving through repeated experimentation that a particular theory is confirmed in reality. To do this, the data and methods used in any one piece of research must be transparent and understood to other researchers. This requires not only publication, typically via a scholarly article, but that other researchers can repeat those steps themselves, through either *replication* or *reproducibility*.

Replication, or redoing the (physical) experiment from scratch, is expensive, and may not be possible due to the passage of time.

Reproducibility typically focuses on the computational aspects of data, combined with code, to produce results, Additionally, funders increasing require data to be made available to others, and the research community expects that data to be prepared in such a way that it is easy for other researchers to use. The days of "contact the author" to request data and code are passing.

The following references provide more context:

Wikipedia on Reproducibility

Science on Replication and Reproducibility

A particularly notable case is described in the Duke "starter set" and article

3.2 Data Management

Good data management is essential for reproducibility. Consult the Rutgers University Libraries Research Guide on Data Management for more guidance from funders on data management and more detail on data management practices. Here we will focus on just a few principles for data preparation:

- Keep raw data pristine and separate from any working data. In the event of any mistakes in data preparation, or later reconsideration of methods, having the original raw data available is essential.
- Document your variables and data collection! Do this for yourself as well as for others. Think about writing about everything you might forget when revisiting the project three years later in response to a query. That will provide clear guidance to other users. It is easier to write this up at the beginning of a project, when the information is fresh in your mind, than to reconstruct it later. This may also help you with organizing the data, such as developing a consistent variable naming scheme and to think through making the data clear and usuable. Disciplinary practices vary, but the documentation can be in the form of a codebook, data dictionary, or a simple readme (the links provide some examples of these).
- If you can, avoid working in Excel, or any other editing environment where you prepare your data manually via point and click. It is very difficult to document the steps taken to clean and filter data in the this environment, and others may not be able to follow your work. Using written code to prepare the data allows others to easily reproduce your steps.
- Store your data, code, and associated files in a standardized directory structure that is easy for others to understand. One tool to automate the creation of such a directory structure is the ProjectTemplate package.

4 Literate Programming

Literate programming is "well-commented" code that explains itself. The concept originated with the book by Donald Knuth. The comment sections should describe what is going on in each section of the code so that someone new to the code can clearly understand what is happening at each stage. Certain environments make this easier. Mathematica notebooks, Jupyter notebooks for Python (and other languages) are examples.

4.1 Literate Programming in R

In R, literate programming can be achieved with LaTeX in combination with Sweave, with *knitr + markdown/Rmarkdown* in RStudio, or with Quarto.

Using these tools:

- we can embed R code and run it as the document is generated
- code that is "tangled" in with text can be extracted, and formatted documents can be "woven" or "knitted" from the literate program
- this always ensures that the latest data and results are actually incorporated
- and helps to document and explain code in context (literate programming)
- PDF, document, and HTML formats are easy to obtain
- The formatR package can clean up your code formatting

4.2 LaTeX and Sweave

While beyond the scope of this workshop, LaTeX remains the most full-featured approach to publication-quality documents of any length and complexity, in which every feature can be controlled in detail, even if sometimes this requires a bit of elbow grease in tracking down and applying codes and styles. The Sweave package allows the use of R code inside LaTeX documents, so that analysis, figures, and tables can be inserted accurately and automatically into the writeup.

4.3 knitr and markdown/RMarkdown

Just as the *markdown* language is a simplified and easy-to-learn version of *html* that still covers most of the major formatting needs that html is used for, *Rmarkdown* is essentially markdown plus the ability to evaluate R code. And the *knitr* package provide a quick yet powerful way to generate documents from Rmarkdown.

- Markdown + knitr has become a popular, lightweight replacement for LaTex + Sweave
- RMarkdown (.Rmd) allows R code execution, math formatting in addition to regular Markdown (.md) functionality
- has simple syntax and implementation
- is well integrated into RStudio (one can also use plain markdown that renders on github natively)
- we can publish documents with one click at RPubs
- and we can always fall back on LaTeX/Sweave for more complex document formatting

See Rmarkdown: the definitive guide and knitr documentation for the full details.

4.4 Quarto

Quarto is a newer, cross-platform approach to generating documents. Importantly, any files in markdown and Rmarkdown will be compatible with Quarto and have no issue with document generation. So switching to Quarto is painless, and offers the benefit of enabling many more options in document and website preparation.

These include the following (quoting the Quarto site):

- **Dynamic Documents**: Generate dynamic output using Python, R, Julia, and Observable. Create reproducible documents that can be regenerated when underlying assumptions or data change.
- **Beautiful Publications**: Publish high-quality articles, reports, presentations, websites, and books in HTML, PDF, MS Word, ePub, and more. Use a single source document to target multiple formats.
- **Scientific Markdown**: Pandoc markdown has excellent support for LaTeX equations and citations. Quarto adds extensions for cross-references, figure panels, callouts, advanced page layout, and more.
- **Authoring Tools**: Use your favorite tools including Positron, VS Code, RStudio, Jupyter Lab, or any text editor. Use the Quarto visual markdown editor for long-form documents.
- **Interactivity**: Engage readers by adding interactive data exploration to your documents using Jupyter Widgets, htmlwidgets for R, Observable JS, and Shiny.
- **Websites and Books**: Publish collections of documents as a blog or full website. Create books and manuscripts in both print formats (PDF and MS Word) and online formats (HTML and ePub).

See the Quarto reference guide for more details

At this point, the live workshop and screencast recording include a brief walkthrough/demo of editing and knitting output in RStudio.

5 Package management

Once we have well-prepared data and well-prepared code, the computational side of reproducibility still deserves attention. We cannot fully reproduce an analysis unless we can verify it computationally. This can be a problem if other versions of software were used for the original work. Often, with commercial software, older versions are simply unavailable over time. As a result, open source software is an important enabler of reproducibility. Anyone can grab copies of the software to execute, and can get older versions if necessary for compatibility.

5.1 Tools for package management

- You can record information about your computing environment (the *sessionInfo()* command in R), letting future users see the exact setup you used
- The renv package helps you create reproducible environments for your R projects. Use renv to make your R projects more isolated, portable and reproducible (replacing *packrat*)
- The Posit Package Manager can be used to access older versions of packages, in combination with the checkpoint package
- The miniCRAN package can be used to create a small, managed repository of selected packages. See "Create a local R package repository using miniCRAN"
- Containerized software are more comprehensive solutions, not R specific, bundling all necessary software into a single, launchable app:
  - Reprozip - containerization, tailored to reproducible research needs
  - Docker - leading general commericical containerization
  - Apptainer (formerly Singularity) - open source equivalent of Docker

5.2 Code sharing and version control

- The same forces (cloud computing, shared platforms, standards) are making collaboration easier than ever
- Github, Gitlab, Bitbucket, and others enable easy collaboration on programming, combined with version control
- these come with significant side benefits for reproducibility due to availability of code
- Rproject in RStudio can integrate with github and other tools, but is specific to RStudio

5.3 continued in Data Publication 2

Check out the Data Publication 2 workshop materials for more on data repositories for data sharing and creating packages in R to share data and code.

6 More publishing tools

We continue with a brief examination of two packages, *blogdown* and *bookdown*, that can help extend your data publication capabilities from single documents to larger, more complex websites and books, while continuing to use the same basic building blocks. Most importantly, these tools let you work with all of the data and statistical functionality of R.

6.1 blogdown

The blogdown package and its associated book allow you to write blog posts/websites that can handle R markdown files natively.

The code below is all that is needed to spin up a basic site.

```
library(blogdown)
dir.create("/home/ryan/R/blogdownsite")
setwd("/home/ryan/R/blogdownsite")
new_site()
serve_site()
build_site()
```

You will want to edit the directory commands to match your local paths. Use *getwd()* if you are not sure of your current working directory. Then create a new subfolder that is **empty** for blogdown to drop the site into. The *new_site()* command will put a sample *hugo* site, structured as a blog, into that folder. The *content* folder is where you would edit to create/modify the posts themselves. The *serve_site()* command builds a preview version of the site and displays it. The *build_site* command creates a publishable version of the site, without previewing it.

6.1.1 Potential issues

This section is only necessary if you are having issues getting your blog to display correctly.

If you need the ./public directory, you may need to manually generate the public folder in the terminal, assuming the working directory is the root of your website repo:

```
hugo -d ./public
```

[From https://github.com/rstudio/blogdown/issues/495

The line below may be a useful addition to your .Rprofile

```
options(blogdown.method = 'html')
```

or just delete your .Rprofile file before building your site, as one post recommends (https://stackoverflow.com/questions/71919138/blogdown-r-markdown-blog-post-doesnt-work)

In fact, this latter method worked best for solving my issues.

6.1.2 Serving the site

To make the site available on the web, the *public* folder generated by the *build_site()* command must be put on a web server. There are many ways to do this that are out of the scope of this workshop.

See this page for more on the full process of setting up a blogdown site

The *blogdown* project is not as active as *bookdown* (below). It relies heavily on hugo, and so making any substantive changes to default layouts and actions will require an understanding of *hugo*. Many users may find it easier to work with *hugo* directly. The one advantage is that *blogdown* will interpret .Rmd files for you. However, one can also process .Rmd files into html using *Quarto* (discussed above), and use those results to generate a blog/website via *hugo*. As is typical in the open source world, you can pick the approach that suits you best.

6.2 bookdown

The bookdown package and its associated book by Yihui Xie allow you to generate book-length documentation, also including data, statistics, and mathematics as needed. This can convert a collection of .Rmd files into a structured html page and also output to PDF, with the .Rmd files becoming chapters. The bookdown package extends the already impressive functionality of Pandoc.

To quote Yihui Xie,

> Merging all chapters into one Rmd file and knitting it is one way to render the book in bookdown. There is actually another way: you may knit each chapter in a separate R session, and bookdown will merge the Markdown output of all chapters to render the book. We call these two approaches "Merge and Knit" (M-K) and "Knit and Merge" (K-M), respectively.

The implications of each approach are discussed here.

We will follow the Get started chapter to provide a quick demo. Click "File -> New Project" in the RStudio menus, and select "Book Project using bookdown". After that, remember to change your working directory match the source files you just had RStudio create. You should also run *library(bookdown)* and *library(rmarkdown)* to be sure that your packages are available. The *render_site()* command will generate the site if the automated preview does not work. The will put a directory called **_book** in your home directory (may not be immediately visible). We can also use the output_format option to generate a PDF, for example.

```
setwd("/home/ryan/R/bookdemo")
library(bookdown)
library(rmarkdown)
```

```
render_site()
render_site(output_format = 'bookdown::pdf_book', encoding = 'UTF-8')
```

Note that Quarto is also building up its book functionality and may supplant bookdown at some point. Both approaches offer a quick publishing option: for Quarto to QuartoPubs, and for bookdown, to the bookdown.org site, but the fact that they generate a standard html directory means that the content can be served in any way you choose. As is typical in the open source world, keeping up with the best approach that combines familiarity, flexibility, adoption, and future-proofing is an ever-changing process as programs leapfrog each other and incorporate each other's features.

The live session and screencast continue with a walkthrough of the book files and building process. Obviously writing and formatting a full book project will be much more time-consuming, but this package hopefully helps ease many of those tasks. The ability to create parallel web and PDF documentation formats is a big plus!

7 Conclusion and continuation

For more information and packages supporting reproducibility, see the CRAN Task View for Reproducible Research

Notably, the **ropensci** project has many packages for reproducibility and collaboration, with many tools that will enhance research productivity.

Check out the Data Publication 2 workshop materials for the continuation of this topic, which covers data repositories for data sharing and creating packages in R to share data and code.

As always, *Enjoy R!*