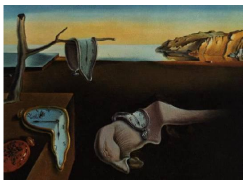

Journal of Statistical Software

January 2014, Volume 56, Book Review 2. *http://www.jstatsoft.org/*

Reviewer: Amelia McNamara
University of California, Los Angeles

Dynamic Documents with R and knitr

Yihui Xie
Chapman & Hall/CRC, Boca Raton, FL, 2014.
ISBN 978-1-4822-0353-0. xxv + 190 pp. USD 59.85.
http://www.crcpress.com/product/isbn/9781482203530

The book *Dynamic Documents with R and* ***knitr***, provides a thorough introduction to both the use and creation of the R package **knitr**. **knitr** is a dynamic document engine, meaning that it produces documents that are responsive to changes in the source code. The most typical use-case for knitr would be embedding R code into an HTML or LaTeX document. A user would write the dynamic source code (e.g., in LaTeX) and use **knitr** syntax to embed R code within it. When the document is processed with knitr, the associated R output (code, data, statistics, plots) is automatically incorporated into the report and updated simply by changing the R code. However, **knitr** is extremely flexible, so that use case should not be considered the only one. Package (and book) author Yihui Xie enumerates many other possibilities, and the extensibility of the package means that you can adjust it for your own needs.

Xie specifies some terminology that I will try to maintain throughout this review; the "dynamic document" is the source code document (e.g., LaTeX and R/**knitr** code) and the "report" is what is produced (for example, a pdf).

knitr was inspired by `Sweave`, a similar tool in base R. `Sweave` was designed for the one specific use case specified above: to embed R code into LaTeX. Xie goes into some detail about the differences between `Sweave` and **knitr** in Chapter 15, but one of the primary differences between the two is **knitr**'s flexibility. You can use **knitr** to embed almost any programming language into any dynamic document. For example, you could include Python code in a Markdown document. In addition, **knitr** comes with more attractive default syntax highlighting, and makes it much easier to modify styles by using "themes," either user-defined or from the package itself (discussed in Chapter 6).

If you are looking to learn how to use **knitr**, this book is for you. There are a limited number of resources for learning **knitr**, because the package is relatively new and the documentation produced by Xie is so good. Between the package website, the associated Google group, and this book, almost any question you might have can be answered. Unlike other cases, where there might be a market for books on the package written by people who are not the package author, I think this book will continue to be the best resource about **knitr**, with the possible exception of the use case I mention below.

Xie says in the preface that the book is broken into two pieces, one aimed at beginners and one for advanced users. Beginners should focus on chapters 1–8, and advanced users might want to skip to chapters 9–11. Before reading the book, I considered myself an “advanced” user of **knitr**, and thought of my undergraduate students (who had not seen Markdown or LaTeX, let alone **knitr**) as beginners. As I read the book, though, I realized that I was an intermediate user who had things to learn from both sections of the book, and the book would probably be overwhelming for someone who had never generated a report from code.

This is more a difference of opinion about what “beginner” means, but I had hoped for some materials that would be appropriate to reproduce for my students as an introduction to the idea of dynamic documents and a quick start guide for new users. Instead, Xie assumes a level of knowledge about typesetting systems and their syntax. For example, on page 2 of the book, Xie makes reference to `\includegraphics{}`, a LaTeX command that I would assume “beginners” would not know about, and a few pages later uses a Monte Carlo algorithm as a “trivial” example. Again, the term beginner in this case should be defined as someone who already uses a typesetting syntax like Markdown or LaTeX, not someone making the transition from pasting R output into Word documents (as my students are). The minimal examples given in Chapter 3 are a nice resource for true beginners, but I think they need a little more context; a place to download the code, instructions on where to open it and what commands to run in order to get the generated report. However, this is my only criticism of the book. Again, I think Xie’s documentation is so clear and thorough that there is little need for any competitors to this book or his website.

The book introduced me to many new ways to use **knitr**, outside of the use case I have already alluded to (embedding R code into LaTeX documents). For example, I was interested to learn that `stitch("yourscript.R")` would allow you to create a report straight from an R source document. Similarly, `spin()` turns R scripts in **roxygen** format into **knitr**/R Markdown reports. If you are using **knitr** within **RStudio** (highly recommended), you can use the notebook button at the upper right of the document pane to quickly create an HTML page from your code (either using `stitch` or `spin`, depending on the format of the comments, it seems). You could also use `knit_rd()` to turn standard R documentation into an HTML website that shows the results from the examples. If you want to use an external code file (rather than including everything inline in one **knitr** document) you can use a commenting syntax similar to **roxygen** to label chunks within your source document, and then reference them in the dynamic document you are coding. All these use cases and more were described in the book, along with simple illustrative examples.

For me, one of the most useful parts of the book is a table, which provides a reminder of start/end and inline syntax for a variety of different typesetting systems. This is useful because it gives a list of possible file formats for the dynamic document, as well as a reminder about how to include R code. I have reproduced it in Table 1.

In Chapter 5, Xie goes into detail about all of these file formats and why you would want to use them. Then in Chapter 13, he enumerates a few more complicated formats that can be used with **knitr**, including HTML5 slides, **Pandoc** (a universal document converter), **Jeckyll**, and **WordPress**. He also explains in Chapter 11 about the other programming languages that are supported with language engines in **knitr**. Language engines give **knitr** the capability to run code in another language and retrieve the results. In the case of languages like C++, code can be run and results reported, and the C++ code can be displayed in the report, as well. With other supported languages like **TikZ** and **Highlight** (a tool for syntax highlighting

Format	Start	End	Inline
Rnw	`<<*>>=`	`@`	`\Sexpr{x}`
Rmd	```` ```{r *} ````	```` ``` ````	`` `r x` ``
Rhtml	`<!--begin.rcode *`	`end.rcode-->`	`<!--rinline x-->`
Rrst	`.. {r *}`	`.. ..`	``:r:`x` ``
Rtex	`\% begin.rcode *`	`\% end.rcode`	`\rinline{x}`
brew			`<\% x \%>`

Table 1: Start, end, and inline syntax for file formats supported by **knitr**.

of code other than R), the goal is typically just to retrieve the results, rather than also display the code.

Also given great treatment are the many ways you can modify and customize **knitr** for your own purposes. The package comes with a variety of pre-built options that parameterize the output from a given chunk of code, including options like `fig.show = 'hold'` to place all plots after the chunk of code, instead of threaded in with it, and `'animate'` to create animations (this requires additional packages). However, if you want to create your own chunk options, you can use any valid R code. For example, Xie shows how to create a chunk option that inserts some particular LaTeX code before and after a chunk. You can also create option templates, which are new named parameters that include a number of standard chunk options. For example, a new option template named `fig.large` that sets both the width and height of a figure, so you do not need to use `fig.width` and `fig.height` every time.

Another common modification that comes up is the width of **knitr** output. My students are often frustrated because the highlighted area of code overhangs the margins of the LaTeX document, sometimes even spilling over the edge of the page itself. Xie explains the possible ways to remedy this problem, namely reducing the width of **knitr** output using `options()`, breaking lines of R code manually (using `paste()`, for example) and making the LaTeX document wider (for example, using the geometry package).

Finally, Xie addresses what I think is the most complicated and fraught piece of **knitr**: caching. Because documents can be long, and re-running the code every time you compile your document is often very time consuming, **knitr** can cache results, plots, etc., so that the output is stored for later use, and only re-run if the code is changed. Although chunks can be given a default caching behavior (and this behavior can be overridden or customized for every chunk) there are still problems that can emerge. Xie explains some of these issues and their treatment.

The most general issue is that while **knitr** uses an MD5 hash to ensure that any change in the code results in an update of the cache, there are changes that are not captured by the hash. For example, if all the code remains the same, but you change the data (for example, updating with a cleaner file or another year's data) that will not be caught by the caching by default. However, you can add additional chunk options to catch modified data. Similarly, while **knitr** makes an effort to cache side effects, they may not all be captured, so it is best to deal with them explicitly. And if chunks depend on one another you should either manually specify dependencies or use the chunk option `autodep`, so that changes in one chunk will result in cache updates to the other chunks that depend on it. Xie also notes that if you are using a language other than R, caching typically does not work (because **knitr** does not know which objects to keep), so dependent code chunks typically need to be combined.

Dynamic Documents with R and ***knitr*** provides a thorough explanation of many different use cases for **knitr**, as well as addressing issues that may arise and how to resolve them. The book is written in Yihui Xie's clear and conversational writing style, which makes it easy to understand what he is talking about, and I do not see another book competing with its content. My only quibble with the book was its lack of "getting started" materials for beginners, but it seems unlikely a true beginner would pay USD 60 for a book if they did not have at least a minimal idea of what it was about. All in all, this is a great read and handy desk reference for the regular **knitr** user.

Reviewer:

Amelia McNamara
Department of Statistics
University of California, Los Angeles
Los Angeles, CA 90095, United States of America
E-mail: amelia.mcnamara@stat.ucla.edu
URL: http://www.stat.ucla.edu/~amelia.mcnamara/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 56, Book Review 2 *Published:* 2014-01-09
January 2014