Austrian Journal of Statistics
2025, Volume 54, 43–49.
http://www.ajs.or.at/
doi:10.17713/ajs.v54i3.2057

Contributions of Fritz Leisch to Vignettes and Reproducible Research

Robert Gentleman
Dana Farber Cancer Institute

Anthony Rossini
UCB and
University of Washington

Vincent Carey
Channing Division of
Network Medicine
Mass General Brigham
Harvard Medical School

Abstract

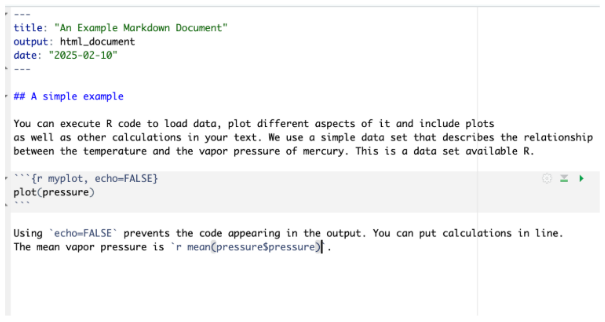
In this paper we review the contributions of Dr. Fritz Leisch to the areas of reproducible research and software documentation. We first review three emergent use cases for these contributions and then describe the computational tools and methods needed to support them. We then discuss some of the risks that exist for each of the use cases and consider developments that might help to mitigate those risks. And we end by considering how they may evolve in the coming years.

Keywords: dynamic document, Sweave, vignette, reproducible research, containerization, R.

1. Introduction

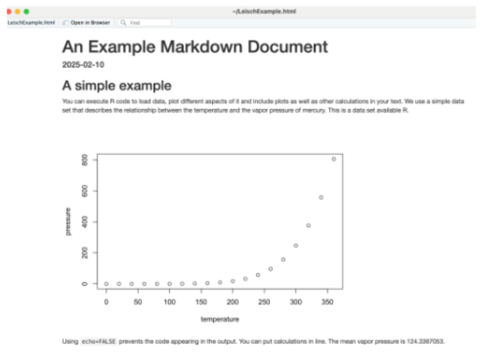
In a pair of papers (Leisch 2002, 2003), Fritz Leisch introduced and described Sweave, a novel synthesis of efforts to document the practice of data science in a transparent and reproducible way. Building on the literate programming proposals of Knuth (1984) and others, an Sweave document explains software usage and includes the source code which carries out computations. The novel aspect of Fritz's contribution was an extension to literate programming in which code segments are executed and the output of the code, potentially tables, graphics, or dynamic content, is embedded in a rendered document along with narration and optionally source code. See Figure 1 for an example. This transforms the idea of literate programming from a method of documenting code to a method of documenting what the code did when applied to a specific problem. A wide range of applications to statistics and data science was originally described by Rossini (2001) and Rossini and Leisch (2003). These ideas can be applied to documentation, verification, explanation, application, and teaching.

Discussions of the opportunities such a system could enable for statistical and data science practice include Rossini (2001), Gentleman and Temple Lang (2007), and Sawitzki (2002). Fritz's great insight was to see that a transformation of the document which inserted artifacts produced by the code, e.g., tables or figures, would lead to improved documentation of software packages and workflows and would simultaneously enable a form of reproducibility of papers, whether for research or teaching. More importantly, he implemented his vision



~~~
---
title: "An Example Markdown Document"
output: html_document
date: "2025-02-10"
---

## A simple example

You can execute R code to load data, plot different aspects of it and include plots
as well as other calculations in your text. We use a simple data set that describes the relationship
between the temperature and the vapor pressure of mercury. This is a data set available R.

```{r myplot, echo=FALSE}
plot(pressure)
```

Using `echo=FALSE` prevents the code appearing in the output. You can put calculations in line.
The mean vapor pressure is `r mean(pressure$pressure)`.
~~~

a) The untransformed document, written in R Markdown



An Example Markdown Document

2025-02-10

A simple example

You can execute R code to load data, plot different aspects of it and include plots as well as other calculations in your text. We use a simple data set that describes the relationship between the temperature and the vapor pressure of mercury. This is a data set available R.

Using `echo=FALSE` prevents the code appearing in the output. You can put calculations in line. The mean vapor pressure is 124.3367053.

b) Shows the rendered version

Figure 1: The two versions of a dynamic document

in a way which enabled others to build on it, thus creating a community of supporters and extenders. As in Gentleman and Temple Lang (2007), we refer to a document that mixes text with computer code in ways that support both extracting the code (tangling) and replacing or augmenting the code with its outputs in a finished form (weaving) as a *dynamic document*.

At approximately the same time that Sweave was being developed, the authors of this paper were engaged in the creation and development of the Bioconductor Project (Gentleman *et al.* 2004). Bioconductor was formed to coordinate and curate software, annotation, and data resources for computational genomic data science, first for the DNA microarray paradigm that emerged at the start of the 21st century, continuing on to single-cell and spatial transcriptomics of intense interest today. Software and workflows in genome-scale biology are inherently complex and documenting them is essential both for ensuring scientific integrity of usage and results, and for building an efficient and reliable workforce. Early in the Bioconductor project it was recognized that Sweave documents could collect and drive execution of examples and demonstrations to efficiently instruct scientists, including those with non-quantitative training, to apply advanced quantitative and informatics methods to the analysis and interpretation of their own work. The requirement that Bioconductor software packages include runable Sweave vignettes was a key factor in the rapid adoption and expansion of the Bioconductor Project. As a by-product, Sweave increased the possibility for reproduction and exploration of experimental data, enabling transparency in a previously opaque analysis setting.

From an informatics and data science perspective, further developments extended and simplified these ideas and tools, ultimately focusing on Markdown (e.g., Xie, Allaire, and Grolemund 2023) as a vehicle for authoring vignettes. The processing steps were modified to support different outputs, such as books, interactive web pages (Shiny; Wickham 2021) and Quarto (Allaire, Teague, Scheidegger, Xie, and Dervieux 2024) thereby creating a very flexible but inherently simple framework which is very popular.

2. Three emergent use cases

The concepts of dynamic documents support three interrelated use cases: 1) documentation of a software package, 2) production and sharing of *How To* manuals and 3) reproducible research papers. Examples of these different approaches are found in many different computing ecosystems. We will use the term descriptive version to refer to the document in the form where code chunks are visible and unevaluated and the term rendered version to refer to the form where the code chunks have been evaluated and used according to the authors' directives.

2.1. Documentation of software packages

It has become common in biology, epidemiology and a number of other disciplines to create software packages that support specific analytic approaches or workflows. In these settings the packages often have a number of different software components that need to work in concert to achieve the desired analysis. It is standard practice in software documentation to provide manual pages for each function describing the inputs, outputs and often optional variations on the standard analysis. Many languages provide ways to document and test each function to ensure its inputs and outputs align with the current documentation. However, this approach may not ensure that the functions work together in the intended fashion.

In this setting a dynamic document can be used to describe the intended workflow, and as such it will function as a *How To* document. In R these documents are called vignettes and they can be checked for validity using tools similar to those used for testing individual functions. Hence a vignette can be both informative and diagnostic. The Bioconductor Project makes substantial use of vignettes for both of these purposes and is a good source of examples.

2.2. *How To* documents

As described above one of the uses of vignettes in R packages is to function as *How To* documents. However, tools like Jupyter and Google Colab notebooks are much more widely used (based on GitHub repository data such as forks, likes and downloads) and play a similar role. Donoho (2024) argues that the ease of use and widespread dissemination of these notebooks has greatly sped up the adoption of new analysis tools. Our own experience with protein folding tools strongly agrees with that observation.

2.3. Reproducible research

There are many different use cases that fit under this heading. One use case is to ensure that all research projects that lead to publications, conference presentations, white papers etc. are reproducible in the sense that there is some documented way of starting with specific data and producing the exact set of figures and tables used in the presentation or paper. Many labs use these methods to ensure that graduate student and post-doc projects are both reproducible and extensible. By extensible we are considering the common situation where a new analyst will extend a previously published study by developing new methods, or alternative approaches. Having access to the actual data, code and workflow that was initially used is of substantial educational benefit and can greatly reduce the time needed to develop new methods.

3. Methods

To make use of dynamic documents in practice there are three technological hurdles that need to be overcome. There needs to be a software development process that will support the code that performs the data manipulations and produces the figures. There needs to be an authoring infrastructure that supports integrating text and code in order to create the descriptive version of the paper and finally there needs to be an infrastructure that can process the document, evaluate the code chunks and appropriately insert the results consisting of tables, figures and other outputs into the final rendered version of the paper. We also note that a number of programming languages have developed tools to create and render such documents, see Table 1. In this section we will outline strategies to address these issues.

3.1. Writing software

The use of web-based platforms such as GitHub (`https://github.com/`) for version control and collaboration has become a standard for software development. Many programming lan-

Table 1: Sweave-inspired extensions for a variety of programming languages

Language	Tool	GitHub stars
Python	Sphinx	6,500
R	R Markdown	2,400
Multiple	Quarto	3,700
Julia	Literate.jl	500

guages have support built in for using these tools and programmers are typically familiar with the tools. These tools are useful for dynamic documents, and provide a well documented resource. It is good practice to consider writing a software package as a way to both encapsulate all the code needed for a dynamic document. This also provides more explicit control of using version numbers to ensure that all authors are using the most recent versions of the code and other components.

These platforms also provide tools for identifying authorship contributions, as well as organizing code and supporting documentation. Recent developments also include methods to execute certain tasks whenever the repository is updated.

3.2. Dynamic documents

Most widely used programming languages support dynamic documents in some form. Perhaps most widely used are notebooks such as those provided by Jupyter. For example, Google's Colaboratory (`https://colab.google/`) uses Jupyter notebooks which can be provisioned with compute resources, including GPUs.

3.3. Processing documents

In order to support the transformations of dynamic documents into different outputs some additional tools are needed. The range of opportunities is quite broad. There can be code written in different languages, data accessed from a variety of sources, computation carried out locally (e.g., on a laptop) or in the cloud or other special GPU based hardware and an author might also want to create a variety of different artifacts rather than a single document. There are also tools for caching outputs to support convenient processing in the presence of long-running computations or data access restrictions.

These challenges can be solved in many different ways. Browser-based or graphical user interfaces employ connections to remote servers that contain the necessary computational engines. Jupyter and Google Colab notebooks as well as Posit's "R Markdown notebook" take this approach.

3.4. Containerization

Tools such as Docker (`www.docker.com`) can address the challenge of accurately describing and sharing the computing environment required to perform the computations. An author can specify a specific operating system, the installed packages and any other resources, such as databases, that might be needed. This recipe is then used to create an image, which is essentially a self-contained piece of software that can be instantiated to create a running machine according to the recipe. The running instance is then called a container. Containers provide a comprehensive method for collaborating, as using a specific container ensures that all collaborators have identical computing environments. Containers are also arguably the most straightforward way to achieve reproducibility, as anyone with access to the container can re-run any analysis that has been documented, relying only on the contents of the container.

3.5. Data access for reproducibility

When authoring a vignette it is common to refer to the data that are needed within the software package that is being documented. In some cases a representative subset of the data may be more appropriate. For *How To* documents it is important to choose data resources that are well supported and easily accessed. Examples include scientific data repositories or public data sets such as NHANES (`https://www.cdc.gov/nchs/nhanes/index.html`) to ensure that users can replicate the examples provided. For published papers it would be helpful to have the data that is used available from a scientific data repository such as GEO (`https://www.ncbi.nlm.nih.gov/geo/`).

4. Risks and mitigations

The primary challenge for reproducible research is the lack of sustainable infrastructure and funding. Scientific journals have not taken the lead in supporting reproducibility, likely due to increased costs without corresponding revenue. Both public and private funders have yet to fully support the necessary computational infrastructure, while professional societies have made limited progress in changing behaviors. Further discussion and examples are provided in a recent commentary on Donoho's manifesto on "frictionless reproducibility" (Barba 2024).

4.1. Functionality risks of dynamic documents

The main concern is that literate or dynamic documents may cease to function as intended over time. While traditional literate programming faced few issues, modern dynamic documents and notebook interfaces present more complex challenges:

- Processing tools may become obsolete.
- Programming languages may evolve, breaking existing code.
- Dynamic documents require specific software versions, hardware, and operating systems.

Currently, there is no comprehensive solution to maintain long-term functionality across all platforms. Mitigation strategies of interest include

Version control: Systems supporting active research languages often include mechanisms for specifying package versions.

Containerization: Using tools like Docker can encapsulate computational needs, allowing for complete specification of the operating system and components.

Cloud infrastructure: Some platforms, like Google Colab, offer tools to create containers that can run on cloud infrastructure.

4.2. Quality assurance in reproducible research

Reproducibility alone does not guarantee quality analysis. While most laboratory sciences have established practices for experimental replication, computational sciences often lack sufficient detail for result replication. This highlights the need for:

- Improved documentation of software and methods.
- Standardized practices for sharing computational workflows.
- Peer review processes that assess both reproducibility and scientific quality.

Addressing these challenges is critical to advance the field of reproducible research and ensure its stability, long-term viability, and credibility.

5. Discussion

The basic ideas of dynamic documents have evolved substantially since Fritz's initial papers. Many important use cases have been developed and some ideas, especially those incorporated in Jupyter notebooks have been widely adopted. Most programming languages used for scientific computing (namely R, Python and Julia) can be used to create and compute with dynamic documents, although it seems that only R has really engaged with the notion of vignettes as critical components of software packages.

Substantial challenges remain for their use in reproducible research. We are not aware of widespread use of these tools or ideas in this context. Software languages evolve over time as does computer hardware. What ran at one point in time may not at a later date, although we believe that principled use of source code repositories and container systems, such as Docker, may ultimately address these concerns. Data is also a concern as not all data can be shared openly due to privacy and other legal constraints, and internet resources have not proven to be particularly stable.

Other hurdles involve the publishing process itself. There is no real evidence that referees can or would make use of anything besides the transformed output and few journals have any mechanism for storing and sharing the necessary files and infrastructure needed.

There are potential solutions to these challenges. Specifically the use of software containers including stable virtual machine infrastructure such as Docker appears to be a reasonably robust way to specify computational environments which can be stored and shared. The wide-spread use of code repositories such as GitHub provides good solutions to software and document development processes. Data sharing or integrity may be helped by ensuring that post publication data are deposited in stable on-line repositories.

There has been true success in the use of vignettes. The Bioconductor Project has many examples of how they are used for teaching, training and as tools to help ensure that internal consistency is maintained in user contributed packages. The use of Jupyter and Google Colab notebooks is widespread, especially in the machine learning community. The ideas Fritz put forth have helped us move forward in meaningful ways. The precise sharing of implementations for for statistical methods and data science, which Fritz's work championed, has made a huge impact in the dissemination of data science techniques and methods, especially in appropriate utilization of those methods for scientific knowledge generation.

References

Allaire JJ, Teague C, Scheidegger C, Xie Y, Dervieux C (2024). "Quarto." doi:10.5281/zenodo.5960048.

Barba L (2024). "The Path to Frictionless Reproducibility Is Still under Construction." *Harvard Data Science Review*, **6**(1). doi:10.1162/99608f92.d73c0559.

Donoho D (2024). "Data Science at the Singularity." *Harvard Data Science Review*, **6**(1). doi:10.1162/99608f92.b91339ef.

Gentleman R, Temple Lang D (2007). "Statistical Analyses and Reproducible Research." *Journal of Computational and Graphical Statistics*, **16**(1), 1–23. doi:10.1198/106186007x178663.

Gentleman RC, *et al.* (2004). “Bioconductor: Open Software Development for Computational Biology and Bioinformatics.” *Genome Biology*, **5**(10), R80. doi:10.1186/gb-2004-5-10-r80.

Knuth DE (1984). “Literate Programming.” *Computer Journal*, **27**(2), 97–111. doi:10.1093/comjnl/27.2.97.

Leisch F (2002). “Sweave, Part I: Mixing R and LaTeX.” *R News*, **2**(3), 28–31. URL https://journal.R-project.org/news/RN-2002-3-sweave-part-i-mixing-r-and-l/.

Leisch F (2003). “Sweave, Part II: Package Vignettes.” *R News*, **3**(2), 21–24. URL https://journal.R-project.org/articles/RN-2003-013/.

Rossini A (2001). “Literate Statistical Analysis.” In K Hornik, F Leisch (eds.), *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, March 15–17, 2001*. Technische Universität Wien, Vienna, Austria. URL https://www.R-project.org/conferences/DSC-2001/Proceedings/Rossini.pdf.

Rossini A, Leisch F (2003). “Literate Statistical Practice.” *UW Biostatistics Working Paper Series*. URL https://biostats.bepress.com/uwbiostat/paper194.

Sawitzki G (2002). “Keeping Statistics Alive in Documents.” *Computational Statistics*, **17**(1), 65–88. doi:10.1007/s001800200091.

Wickham H (2021). *Mastering Shiny: Build Interactive Apps, Reports, and Dashboards Powered by R*. 1st edition. O’Reilly Media, Beijing Boston Farnham Sebastopol Tokyo. URL https://mastering-shiny.org/.

Xie Y, Allaire JJ, Grolemund G (2023). *R Markdown: The Definitive Guide*. Chapman & Hall/CRC. URL https://bookdown.org/yihui/rmarkdown/.

Affiliation:

Robert Gentleman
Department of Data Science
Dana Farber Cancer Institute
Boston, MA, USA

Austrian Journal of Statistics http://www.ajs.or.at/
published by the Austrian Society of Statistics http://www.osg.or.at/
Volume 54 *Submitted:* 2024-10-03
2025 *Accepted:* 2025-03-07