Scientific documents with Quarto

COUSIN Workshops Series

Felipe Ortega María Jesús Algar

2024-12-20

Table of contents

Preface

This workshop describes how to use Quarto, software for producing scientific documents and publications, in ecology and plant research.

Quarto is a powerful and versatile tool for researchers implementing **reproducible** workflows. The quest for open-access research, including the final product (manuscripts) and ancillary research materials like source code, datasets, figures, pipelines or setup files, has become a prominent concern among scholars and practitioners in many fields. Prestigious publications require authors to submit these materials alongside manuscript drafts to let other colleagues reproduce and validate the results, replicate studies in new cohorts or improve their interpretability.

Quarto combines formatted text and executable source code chunks into a single document. Code *chunks* can be written in different programming languages such as R, Python, Julia or Observable. As we will see, it is possible to combine different programming languages in the same document or collection of documents, increasing the flexibility of this tool.

This is a **practical guide**, presenting hands-on examples and code to produce your own Quarto documents quickly. In addition, key concepts and best practices are also presented to steer new Quarto apprentices in the right direction.

To learn more about Quarto visit the comprehensive guide. Quarto can produce standalone documents, books like this one, as well as complete websites.

Part I

Quarto

1 Scientific documents

In their daily work, students, academics and scientific specialists produce a large amount of documentation of all kinds: laboratory notes, lectures, memos, technical reports and, above all, scientific articles to publish their discoveries and advances in an area of knowledge. Normally, the creation of this type of scientific documents involves a large number of tasks involving different tools and possible points of failure.

Figure 1.1 shows a schematic overview of a classic workflow for creating scientific documents. The main element is often a word processor master file (Word, OpenOffice/LibreOffice, etc.), a web page, or a LaTeX file (if we are creating a PDF document) that holds all contents.

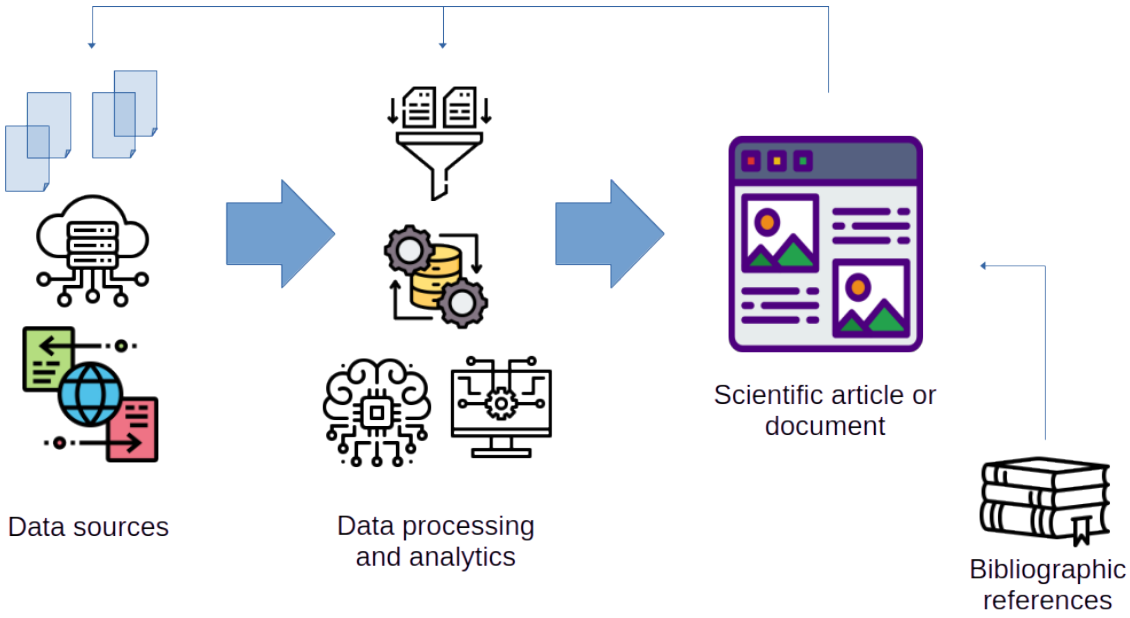

Figure 1.1: Creation process for cientific articles and documentation.

This master file is filled with content from a variety of sources, such as:

- figures and diagrams generated manually or through software code (such as data visualization charts);
- tables and summaries describing data sets and results;
- results and evaluation of the performance of models or algorithms; statistical or machine learning;
- mathematical formulas and equations;
- data tables and other useful information;
- bibliographic references (usually generated with the help of some bibliographic information management program).

Many of these elements force users to run external tools and programs, procedures, and other tasks over and over again to then incorporate the new results into the master file. We must admit that

this process, which is mostly manual, is not only tedious but also very prone to errors or oversights. "Wait! I forgot to update Figure 1." "Are you sure these are the latest evaluation results for model *M*?" "Have you checked that we have uploaded the latest version of the data file *D*?" These are common questions that arise in the day-to-day work of scientific teams.

However, it would be great if it were not necessary to carry out all this manual and sometimes very frustrating process manually. Do we have any alternative to avoid it? Yes, we do. The answer to our needs is provided by a very powerful concept: **literate programming**.

1.1 Literate programming

The concept of literary programming was coined by Professor Donald E. Knuth (1984). Yes, you read that right, more than 40 years ago. This concept states that it should be possible to integrate, in a single scientific document, formatted text and results of the execution of software code to compose said document dynamically. So, why has it taken us so long to put this idea into practice? Knuth's vision, although very ahead of its time, was correct, but the technology of the time did not allow it to be put into practice.

However, today we have all the essential elements to make it real. What's more, we have a tool, Quarto, that lets us automate and manage the whole process of creating literary programming documents quickly and reliably.

1.2 Reproducible research

For many decades, the scientific method has been based on the publication of research papers describing the results of data analysis and experiments. In all cases, it is essential to be able to trust the conditions, the data collected, the method of analysis and execution of the experiments, as well as the various kinds of tools, including software, that the authors of the publication used to carry it out.

However, the numerous advances in recent years in the tools and methods of analysis make it much easier to check the results of these analyses. We might assume that this makes the work of scientists much easier, but in reality the opposite is true. Let us look at some examples:

- **Oncology** (Begley & Ellis, 2012): The Biotechnology Department of the firm Amgen (Thousand Oaks, CA, USA) was able to confirm only 6 of a total of 53 emblematic research articles published in this area. Bayer HealthCare (Germany) was able to validate only 25% of the studies analyzed.
- **Psychology** (Wicherts et al., 2006): 73% of the authors of a total of 249 articles published by the APA did not respond within a period of 6 months to the questions and requests formulated about the data they used in their research.
- **Economics and Finance** (Burman et al., 2010): A comparison of different software packages applied in the execution of various financial and statistical model analyses shows that each of these packages produces *very different* results using *the same statistical techniques* directly applied to *identical data* as those used in the original publication.

In fact, articles have even appeared suggesting that many of the results published in areas such as Medicine may not be entirely reliable (Ioannidis, 2005). As a result of all these recent findings, a great controversy has been generated throughout the scientific and research community, accompanied by a deep crisis of confidence.

Nevertheless, as a well-known comic strip about the academic world and research (see Figure 1.2) very well describes, the process of developing scientific publications is based primarily on the continuous review of methods and results (starting with the students themselves and their supervisors).

The Figure 1.3 shows a graph published in the prestigious journal Science Magazine (Brainard et al., 2018), which represents the data on the evolution of the number of research articles retracted or withdrawn for various reasons, between 1997 and 2014. In this graph, we can see how the improvement of tools and the greater availability of resources allow for the analysis and review of a greater volume of publications and analyses, which allows for the detection of a greater number of problematic cases.

1.2.1 Reproducibility and replicability

There is often talk of *reproducing* and *replicating* a data analysis or a scientific experiment (Leek & Peng, 2015). However, many evidences can found showing that there are incompatible definitions of these two and other related terms (Barba, 2018). Be very careful, therefore, because depending on the scientific community or the field of knowledge in which we find ourselves, the meaning of these two terms may even be *entirely opposite* to their accepted definition in other areas [1]. Here we will stick to the definition accepted in a large number of areas, including statistics or scientific computing (see Barba, 2018, p. 33):

- **Reproducibility**: It is defined as the ability to recompute the results of an analysis, with the same data that were used in the original analysis, and knowing the details of the sequence (*workflow* or *pipeline*) of operations that make up said analysis. Certain premises must be able to be guaranteed:
- If we use the same tools (e.g. R, a certain list of packages, the same versions of all packages and dependencies), as well as the same code (*R scripts*) on the same data, the results and conclusions must be consistent with those of the original analysis.
- The authors of the original analysis must provide all the elements (data, code and procedure used) to allow the analysis to be reproducible (Barba, 2018).
- **Replicability**: It is defined as the ability to perform an experiment or analysis independent of the original, that addresses the same objective but on a set of data different from that used in the initial study. If the results are not consistent, it will be necessary to carry out more replications and harmonize the results and conclusions through appropriate techniques, such as **meta-analysis**.

[1] Among the most important examples of definitions that contradict those we give in this workshop are those adopted by the Federation of American Societies for Experimental Biology (FASEB), in immunology and microbiology, as well as those adopted by the Association for Computer Machinery (ACM) in computer science.

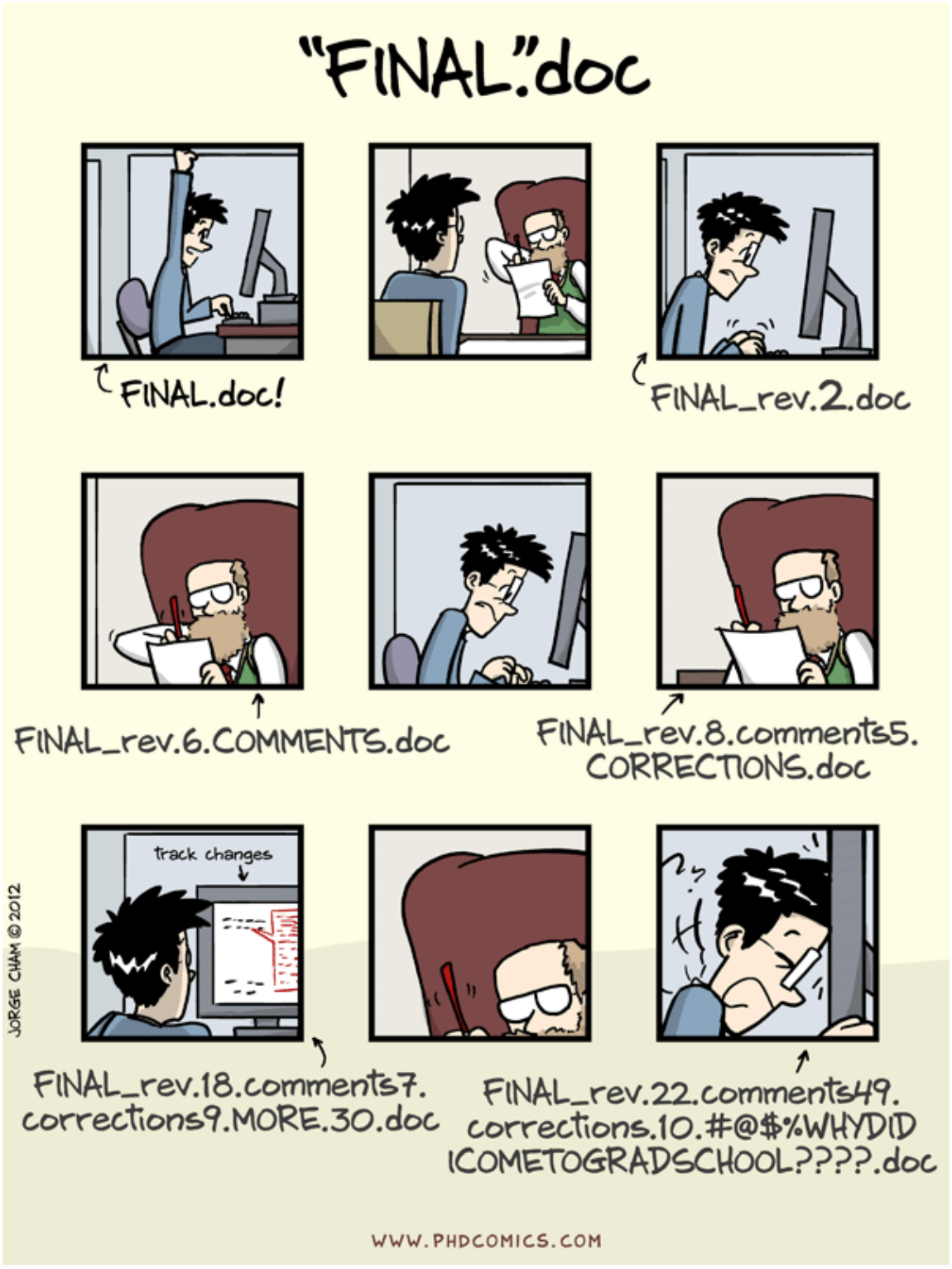

Figure 1.2: Comic strip representing the review model for scientific publications. Source: PhD comics.

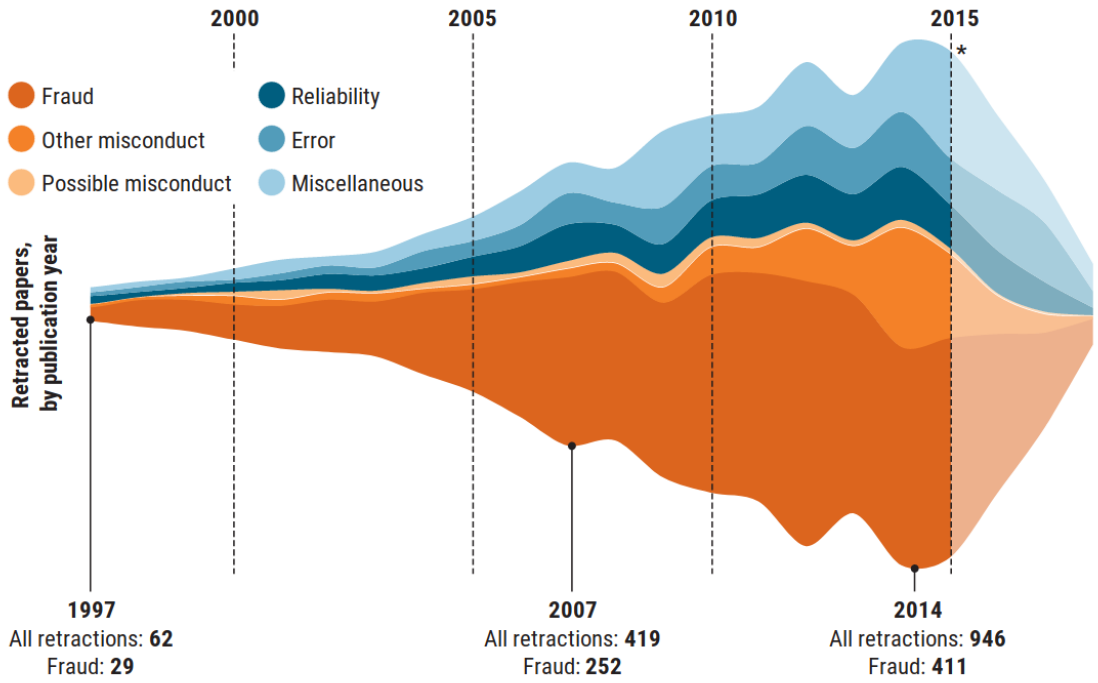

Figure 1.3: Evolution of the number of scientific publications retracted or withdrawn for various reasons, between 1997 and 2014. Source: Science Magazine (Brainard et al., 2018).

1.2.2 Replication levels

Depending on the elements published by the authors of the original study, as well as the level of detail with which the process for carrying out the study is described, the steps that have been followed and the tools used, we have different levels of replicability or reproducibility, represented in the **?@fig-spectrum-replica**.

- *Not reproducible*: No data, code or any specific description of the implementation of the study or analysis is provided. Many scientific publications no longer accept publishing articles under these conditions.
- *Code* or *Data*: A good number of publishers request that the data sets used in the analysis or study of the publication be accessible through a URL, either because they are available in a public repository or because the authors of the article have published it. Likewise, many publications require that the software code to carry out the analysis is also publicly accessible, in an open source repository or in a freely accessible version control service project.
- *Code and data*: Ideally, both the code and the data should be publicly accessible for anyone who wants to examine them or use them to reproduce the results (validation) or replicate the analysis with other data or other cases.
- *Runtime environment and linked data*: A further step to facilitate the reproducibility of studies consists of publishing code and metadata files with more precise information about the programming language, the software packages used and any other dependencies necessary to carry out the same study or analysis. Another variant to facilitate reproducibility is to encapsulate the code and dependencies in a preconfigured virtual container, which can be downloaded and executed directly.

- *Gold standard*: The most advanced level would consist of documenting all the procedures performed during the study or analysis, including the coding of the tasks of obtaining, cleaning and preparing the data, as well as the generation of graphics to visualize the results or any other results derived from the study.

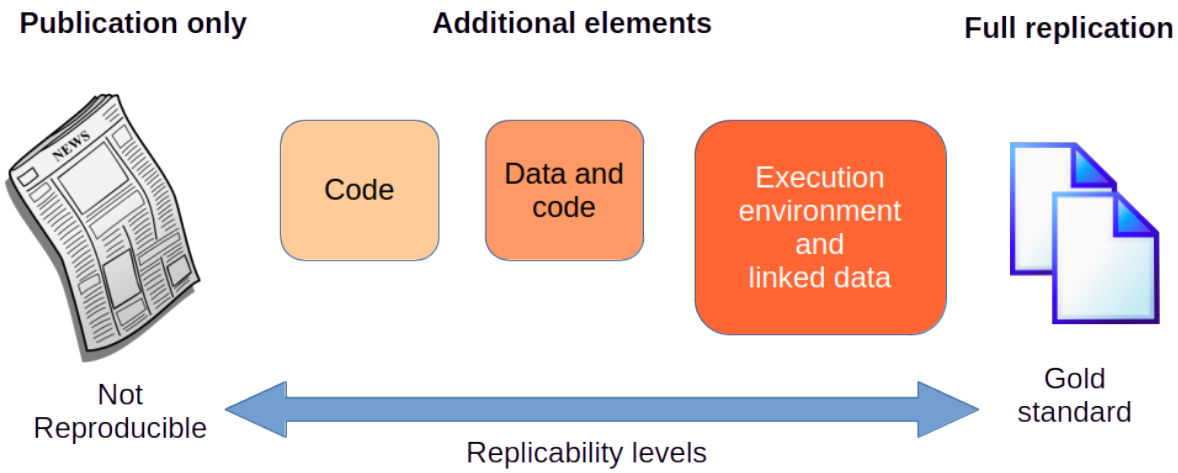

Figure 1.4: Replication levels spectrum in scientific publications. Source: Peng (2011).

1.2.3 Replicability tools

Certain technologies and tools that have become more sophisticated and refined in recent years are making it easier to replicate data processing and analysis.

- Version Control Systems for software code (SCV): tools such as Git, Mercurial and web services such as GitHub or GitLab have popularized the creation and publication of projects that allow the management of the software code that has been created, controlling the changes and the released versions. Web services also integrate a good number of tools to support different facets of the software development process, such as the generation of documentation, manuals and examples, error reports and requests for improvements, continuous integration and continuous deployment (CI/CD), systematic testing of the generated code, etc. If you have not yet considered how using a source code version control tool can benefit you, take a look at Figure 1.5 where you will relive a situation that is unfortunately very common among researchers and scientists who develop software solutions.

- Software virtualization and containers: In a technological environment dominated by the contracting and deployment of computing infrastructure and services in cloud computing architectures, packaging and virtualization tools for software applications and services that can be installed and deployed in a short time have revolutionized the way software products are published and managed, including data processing and analysis products.

- Data version control: In a similar way to SCV for source code, software is appearing to apply the same principles to data files. In this way, we can control different versions of each data file, modifications made to them, etc. One of these tools is Data Version Control (DVC), which allows versioning of data and models. As a result, we can know at all times which version of the data and which list of *features* have been included in each model considered during the analysis, keeping the descriptive information about these three essential components that must always be cohesive.

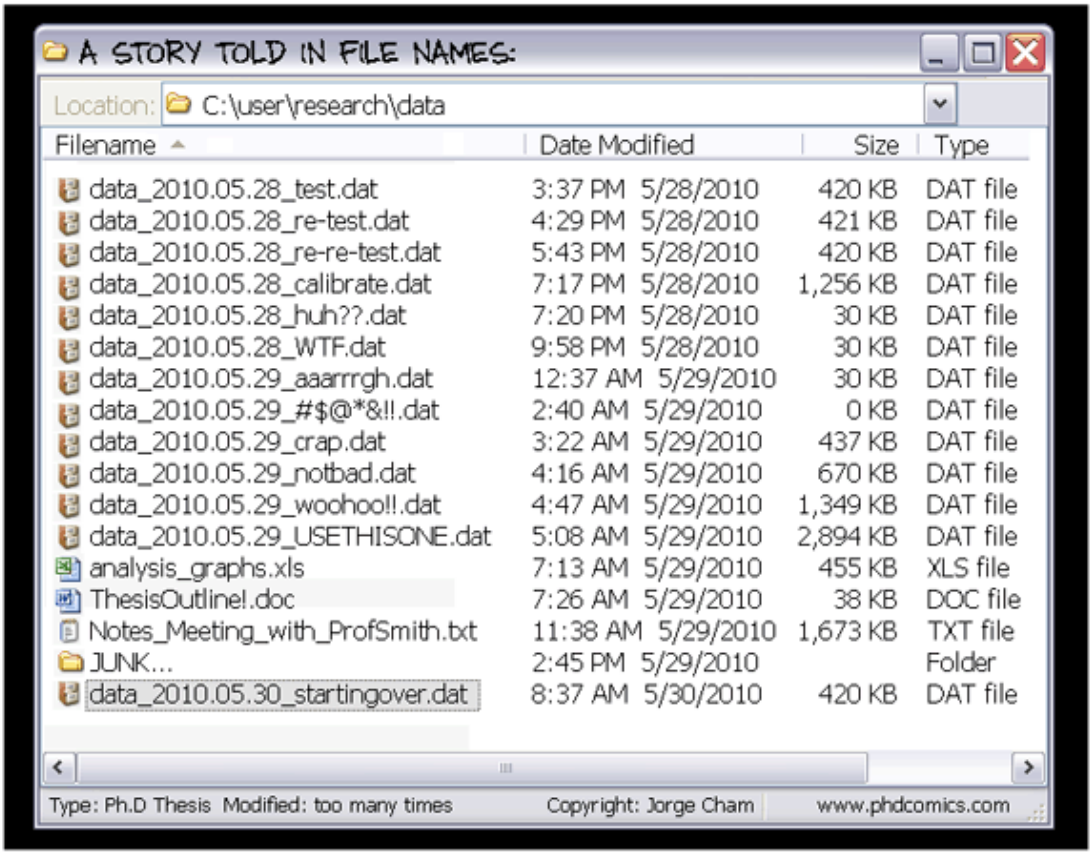

Figure 1.5: Software version control. Source: PhD Comics

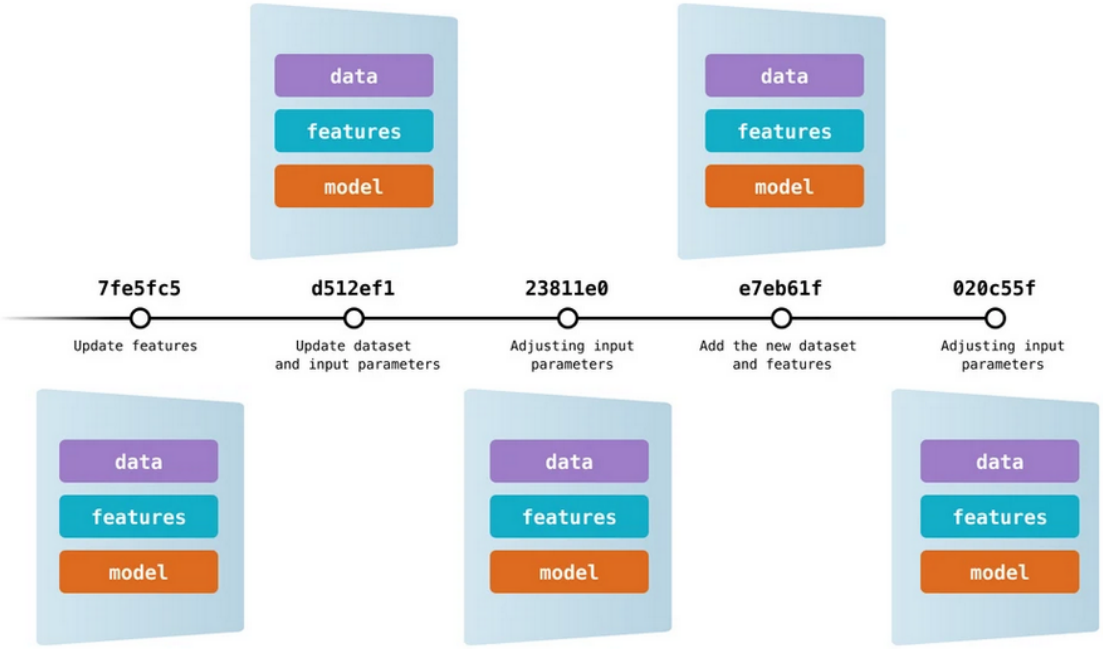

Figure 1.6: Data, code and model versioning example maintained by DVC. Source: DVC Documentation.

- Model and experiment management: Another type of machine learning project management tool is one that allows the organization, monitoring, comparison, and selection of the experiments and models we have carried out. One of the most recent notable examples is ML Flow, which provides support for model tuning, evaluation, and optimization, deployment of models in production environments, creation of a registry of pre-trained models, etc. Of course, it is possible to combine this type of tool with others such as DVC, creating a comprehensive management environment for our projects.
- Creation and management of data processing pipelines: the last essential element in any data processing and analysis project that must take care of scalability is a tool for creation and management of data processing and analysis flows or pipelines. The set of all the pipelines in our project make up the general workflow of the project. These tools are known as data or workflow orchestrators. In this category, we have both very powerful and feature-packed tools such as Apache Airflow or Prefect and simpler and more straightforward ones such as Luigi.

Of course, the R community has not remained oblivious to these new trends, in particular the R OpenSci initiative, within which we find many packages (published in the official CRAN repository) that cover various aspects of scientific work, including the management of *pipelines* and *workflows* through the `targets` package.

- User manual for the R package `targets`.

1.3 Quarto for scientific publications

Now that we know the fundamental concept on which Quarto works and its application to achieve a higher level of reproducibility and transparency in our scientific process, we are going to explain in more detail the process that Quarto follows to compose a document. The Figure 1.7 presents a diagram with the document creation process and the elements and tools that come into play to achieve it.

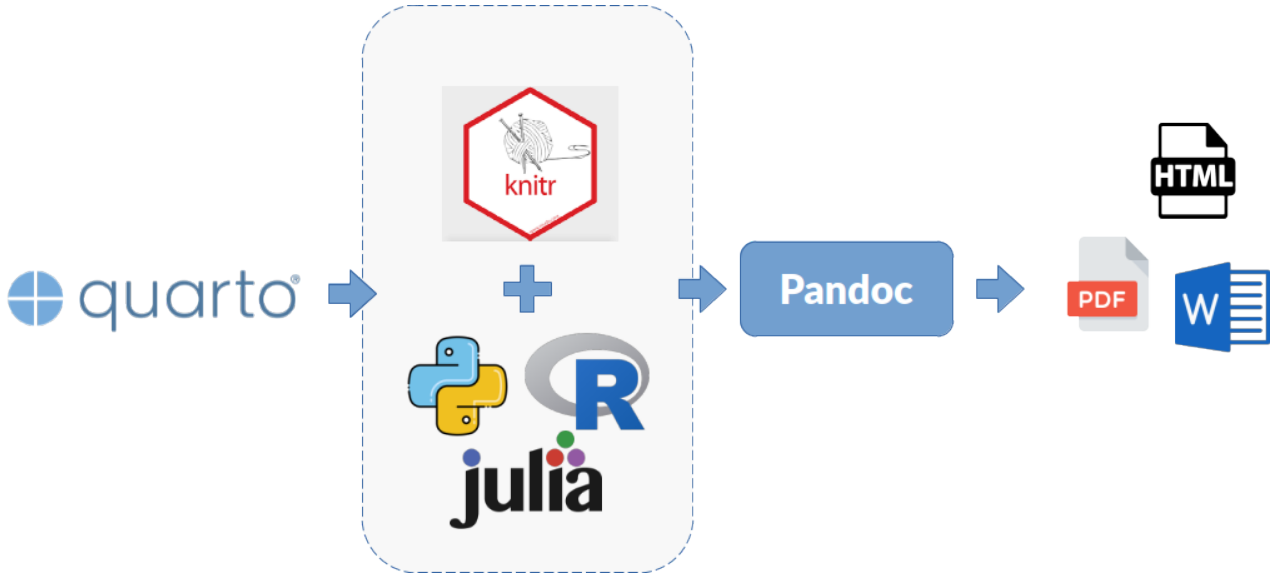

Figure 1.7: Content creation process with Quarto.

- **Quarto**: a software that allows you to create scientific documentation following the principles of literary programming.
- **Knitr and programming language**: the `knitr` package is responsible for the connection with an interpreter of a programming language (R, Python, Julia) that can be executed in a REPL environment, in order to be able to execute software code fragments integrated into the document and generate content in Markdown format as a result.
- **Markdown** (formatted content): a textual content markup language that allows easy formatting of the information in our documents created with Quarto.
- **Pandoc** (universal translator of document formats): this software receives the content already formatted using the Markdown standard, to convert it into the selected output type. There are several options available: HTML, PDF or Word, as well as slides, websites or interactive panels (*dashboards*).

1.4 Quartion installation

To install the latest version of Quarto software on your system, point your web browser to the page https://quarto.org/docs/get-started/. Here, download and install the file corresponding to your operating system.

At this time, the latest version of Quarto available is `1.5.57`.

> **i** Software requirements to generate PDF documents
>
> By default, the output format of documents generated with Quarto is HTML. If we want to generate PDF documents, we need to have a LaTeX distribution installed. For more information, see **?@sec-pdf-reqs**.

2 Document types

2.1 Individual documents

2.2 Books

2.3 Manuscripts (scholarly articles)

2.4 Presentations

2.5 Websites

2.6 Dashboards

3 Quarto workflow

3.1 Required tools

3.2 Producing HTML

3.3 Producing PDF

3.3.1 Customising PDF documents

4 Single Documents

4.1 Preamble

4.2 HTML output

4.3 PDF output

4.4 Producing documents

4.4.1 Preview

4.4.2 Selecting output type

4.4.3 Further Customisation

Part II

Quarto books

5 Creating books

5.1 Authoring tools

5.2 Reference management

5.3 Customisation and templates

6 Creting a field guide

6.1 Templates

6.2 Project management

6.3 Publishing

Part III

Publications

7 Scientific articles

7.1 Replicability in scientific publications

7.2 Publication-ready figures

7.3 Journal article templates

7.4 Examples and best practices

8 FAIR data principles

8.1 Overview

8.2 Source code publication

8.3 Dataset publication

8.4 Reference management

9 Additional resources

In summary, this book has no content whatsoever.

See Knuth (1984) for additional discussion of literate programming.

References

Barba, L. A. (2018). Terminologies for reproducible research. *arXiv Preprint arXiv:1802.03311*.

Begley, C., & Ellis, L. (2012). *Drug development: Raise standards for preclinical cancer research. Nature.[Online]. 483 (7391)*.

Brainard, J., You, J., et al. (2018). What a massive database of retracted papers reveals about science publishing's "death penalty." *Science*, *25*(1), 1–5.

Burman, L. E., Reed, W. R., & Alm, J. (2010). A call for replication studies. *Public Finance Review*, *38*(6), 787–793.

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, *2*(8), e124.

Knuth, D. E. (1984). Literate programming. *Comput. J.*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Leek, J. T., & Peng, R. D. (2015). Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, *112*(6), 1645–1646.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226–1227.

Wicherts, J. M., Borsboom, D., Kats, J., & Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *American PAsychologist*, *61*(7), 726.

A Code reference

A.1 Quarto statements

A.2 R statements

B Integrated Development Environments for Quarto

B.1 R Studio

B.2 Visual Studio

B.3 Positron

C Useful R packages

C.1 Ecology

C.2 Data visualisation

C.3 Data processing

C.3.1 Tidyverse

C.3.2 Alternatives to the Tidyverse

C.3.3 Pipelines

C.4 Spatial data

C.4.1 `sf` (Simple Features)

C.4.2 `terra`

C.5 Time series

C.5.1 Tidyverts

C.6 Data visualisation

C.6.1 `ggplot2`

C.7 Data analysis and Machine Learning

C.7.1 Tidymodels

C.7.2 mlr3

D Producing PDF documents

D.1 PDF documents with Quarto

D.2 Quick LaTeX primer

D.3 Available templates

References

Barba, L. A. (2018). Terminologies for reproducible research. *arXiv Preprint arXiv:1802.03311*.

Begley, C., & Ellis, L. (2012). *Drug development: Raise standards for preclinical cancer research. Nature.[Online]. 483 (7391)*.

Brainard, J., You, J., et al. (2018). What a massive database of retracted papers reveals about science publishing's "death penalty." *Science*, *25*(1), 1–5.

Burman, L. E., Reed, W. R., & Alm, J. (2010). A call for replication studies. *Public Finance Review*, *38*(6), 787–793.

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, *2*(8), e124.

Knuth, D. E. (1984). Literate programming. *Comput. J.*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Leek, J. T., & Peng, R. D. (2015). Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, *112*(6), 1645–1646.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226–1227.

Wicherts, J. M., Borsboom, D., Kats, J., & Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *American PAsychologist*, *61*(7), 726.