Rχiv-Maker: An Automated Template Engine for Streamlined Scientific Publications

Bruno M. Saraiva[1,✉], **Guillaume Jaquemet**[2,3,4,✉], **and Ricardo Henriques**[1,5,✉]

[1]Instituto de Tecnologia Química e Biológica António Xavier, Universidade Nova de Lisboa, Oeiras, Portugal
[2]Faculty of Science and Engineering, Cell Biology, Åbo Akademi University, Turku, Finland
[3]InFLAMES Research Flagship Center, University of Turku, Turku, Finland
[4]Turku Bioscience Centre, University of Turku and Åbo Akademi University, Turku, Finland
[5]UCL Laboratory for Molecular Cell Biology, University College London, London, United Kingdom

Modern scientific publishing has moved towards rapid dissemination through preprint servers, putting greater demands on researchers for preparing and quality-checking manuscripts. We introduce RXiv-Maker, a comprehensive system native to Github that simplifies scientific writing through markdown-based authoring with automated LaTeX conversion. It's specifically designed to help produce preprints for curation in arXiv, bioRxiv, and medRxiv. The system lets researchers write in the familiar and lightweight markdown syntax while generating publication-quality documents automatically. RXiv-Maker offers flexible compilation strategies, including cloud-based GitHub Actions, interactive Google Colab notebooks, and reproducible local builds via Docker containerisation, ensuring consistent environments and eliminating dependency conflicts. The framework inherently supports reproducible research by enabling programmatic figure generation using Python libraries and script-based diagramming with Mermaid.js. This self-documenting article, created entirely within the framework, shows how this markdown-centric workflow transforms scientific communication into an efficient, collaborative, and transparent process, empowering researchers to focus on content while upholding rigorous standards of quality and reproducibility.

article template | scientific publishing | preprints

Correspondence: ***(B. M. Saraiva) b.saraiva@itqb.unl.pt; (G. Jaquemet) guillaume.jacquemet@abo.fi; (R. Henriques) ricardo.henriques@itqb.unl.pt***

Main

Today's scientific landscape is marked by the swift sharing of research findings, a trend largely driven by the exponential growth of preprint servers like arXiv, bioRxiv, and medRxiv (1–3). This shift, while accelerating scientific discovery and enhancing research transparency (4, 5), puts a significant new burden on researchers to produce polished manuscripts frequently and efficiently. Traditional tools and workflows for scientific writing, often reliant on proprietary word-processing software, are poorly suited to this new reality. They pose major challenges for version control, collaborative authoring, and ensuring the computational reproducibility of the final document (6, 7). To tackle these systemic issues, we've developed RXiv-Maker, a GitHub-native framework designed to streamline the entire scientific writing and publication pipeline. The system is built on the principle of using markdown, a simple and intuitive plain-text formatting syntax, as the primary authoring language. This approach fundamentally separates the scientific content from its

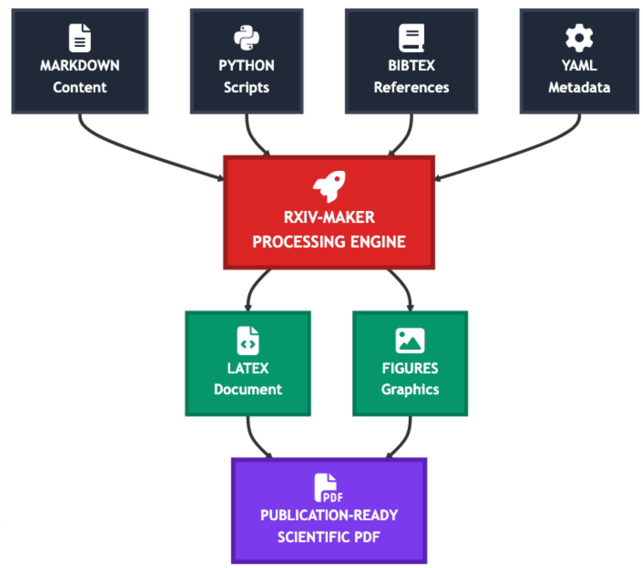


Fig. 1. The RXiv-Maker Diagram. The system integrates Markdown content, YAML metadata, Python scripts, and bibliography files through a processing engine. This engine leverages Docker, GitHub Actions, and LaTeX to produce a publication-ready scientific article, demonstrating a fully automated and reproducible pipeline.

final presentation, allowing researchers to focus on the substance of their work. The plain-text nature of markdown files makes them ideally suited for management with version control systems like Git. This integration provides an unparalleled level of transparency and traceability for the evolution of a manuscript. Every change, every contribution, and every revision can be precisely tracked, attributed, and, if necessary, reverted. Furthermore, it resolves the convoluted and error-prone process of merging changes from multiple collaborators, a common bottleneck in conventional workflows that often leads to duplicated effort and loss of information.
At the heart of the RXiv-Maker philosophy is the pursuit of genuine reproducibility, a fundamental aspect of scientific integrity that extends beyond the experimental data to encompass the entire publication process (8–10). Our framework embodies this principle by enabling the programmatic generation of figures and tables, following best practices for computational research (11). Rather than manually inserting static image files, which obscures the connection between the data and its visualisation, our system promotes the use of scripting languages like Python, with its powerful data visualisation libraries such as Matplotlib (12) and Seaborn (13). Figures are generated directly from the source data and anal-

ysis scripts during the manuscript compilation process. This creates an unbreakable, auditable chain from raw data to the final figure. If the underlying data is updated or the analysis is refined, all affected figures are automatically regenerated, ensuring complete consistency and eliminating the possibility of outdated visuals persisting in the manuscript. This dynamic approach transforms figures from mere illustrations into reproducible scientific artefacts. The system also integrates Mermaid.js (14), a tool for generating complex diagrams and flowcharts from a simple, text-based syntax. This is particularly valuable for creating clear, version-controlled illustrations of experimental workflows, conceptual models, and algorithms, which are essential for communicating complex ideas in many scientific disciplines. RXiv-Maker, therefore, offers a holistic solution that treats the manuscript not as a static document but as the executable output of the research itself.

RXiv-Maker addresses these requirements through a markdown-centric authoring system that automatically translates familiar markdown syntax into professional LaTeX documents. Built upon the established HenriquesLab bioRxiv template (15), the system extends capabilities through automated processing pipelines, integrated figure generation, and flexible deployment strategies. The architecture, detailed in Fig. 1 and comprehensively illustrated in Sup. Fig. S1, provides automated figure generation for statistical visualisation, integrated Mermaid diagram creation, and robust build automation through containerised environments. The technical details of the figure generation system are described in **??**.

Using the RXiv-Maker framework results in a highly efficient and robust workflow for producing professional-quality scientific papers. The system's primary output is a fully typeset PDF document, as seen in the article you're currently reading, which was generated entirely using this process. The markdown source files are automatically converted into a structured LaTeX document, then compiled to produce a PDF with a clean, academic layout, proper pagination, and high-resolution figures. Bibliographic management is handled seamlessly through integration with a standard BibTeX file. The system automatically processes this file to generate correctly formatted in-text citations and a comprehensive bibliography section according to a specified citation style. This automation eliminates the tedious and error-prone task of manually formatting references.

Its utility is further highlighted by the system's flexible deployment options, which cater to a broad range of user preferences and technical environments. We have successfully validated three distinct compilation pathways. Firstly, the cloud-based GitHub Actions workflow offers a fully automated, hands-off experience. Every time code is pushed to the repository, the action is triggered, building a Docker container with all necessary dependencies, compiling the manuscript, and releasing the resulting PDF as a downloadable artefact. This continuous integration pipeline ensures a current, correctly compiled version of the manuscript is always available, serving as a top-notch quality control mechanism for

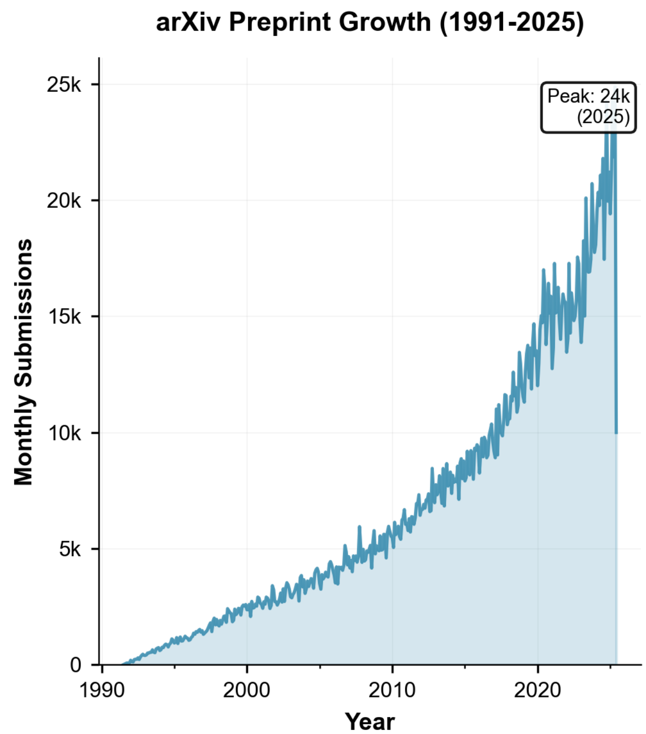


Fig. 2. The growth of preprint submissions on the arXiv server from 1991 to 2025. The data, sourced from arXiv's public statistics, is plotted using a Python script integrated into our RXiv-Maker pipeline. This demonstrates the system's capacity for reproducible, data-driven figure generation directly within the publication workflow.

collaborative projects. Secondly, for users who prefer an interactive, web-based environment, the system can be deployed in a Google Colab notebook. This method removes all local software requirements, allowing users to compile the manuscript simply by executing cells within the notebook, making the framework exceptionally accessible to those with limited command-line experience.

Thirdly, for local development, the Docker-based approach provides a perfectly reproducible compilation environment on any host machine. By running a simple command, a user can start the build process within a self-contained Docker container that encapsulates the exact versions of LaTeX and all other required system libraries. This completely eliminates the 'works on my machine' problem, guaranteeing that the output is identical byte-for-byte regardless of the user's local operating system or software configuration (16). The integration of programmatic figure generation was also validated, supporting interactive computational environments like Jupyter notebooks (17). Python scripts placed within the designated directory were automatically executed during compilation. These scripts loaded data, performed analyses, and generated visualisations, which were then saved as image files and seamlessly included in the final PDF. Similarly, Mermaid.js diagrams embedded within the markdown source were correctly rendered into SVG images and incorporated into the document. This programmatic integration demonstrates a closed loop of reproducibility, where the final manuscript serves as a verifiable and self-contained record of the research findings and their presentation.

Our RXiv-Maker system marks a major step forward in how scientific manuscripts are prepared. By using plain-text markdown and robust open-source tools like Docker, we've created a workflow that boosts efficiency and promotes best practices for reproducibility and collaboration. The main benefit of our framework is that it removes technical complexity from the author's hands. Scientists can focus on the research content and narrative, using simple and widely understood markdown syntax, while the system handles the complex and often frustrating aspects of typesetting, reference management, and dependency control. This approach embraces literate programming principles (18), creating documents that seamlessly blend narrative text with executable code. This is a significant departure from traditional workflows, where researchers often have to act as amateur typesetters, spending hours wrestling with the formatting intricacies of word processors or raw Latex. The integration with Git provides a robust platform for collaborative writing (19, 20), far superior to the chaotic exchange of files via email. It enables transparent attribution, conflict-free merging of contributions, and a complete, auditable history of the manuscript's development.

Within the larger context of the open science movement, RXiv-Maker acts as a practical tool for turning principles into reality. By focusing on automated figure generation and a fully containerised, reproducible build process, the path from raw data to final publication becomes transparent and verifiable. This directly tackles the 'reproducibility crisis' by making it easy to create publications that are not just reports of research, but are themselves computationally reproducible artefacts. This aligns with a growing consensus that the publication itself should be accompanied by the code and data needed to reproduce its findings (8). Our framework makes this possible by design, treating the manuscript and the code to generate it as two sides of the same coin. While other platforms and tools for scientific writing exist, including sophisticated environments like DL4MicEverywhere (21), RXiv-Maker stands out through its simplicity, flexibility, and tight integration with the GitHub ecosystem, which is already the de facto standard for collaborative software development and is increasingly used for scientific projects.

Although Rxiv-Maker has its advantages, we acknowledge certain limitations and areas for future improvement. The current system is mainly designed to produce PDF outputs using Latex. While this is the standard for many scientific disciplines, future versions could support other output formats, such as HTML for web-native articles, potentially leveraging universal document converters like Pandoc (22). Additionally, concerns about reporting quality in preprints (23) suggest opportunities for integrating automated quality checks. Additionally, although the system is designed to be accessible, researchers new to Git and markdown may encounter an initial learning curve. To address this, we plan to develop more comprehensive documentation and tutorial materials. Future work will also focus on deeper integration with data analysis environments like Jupyter notebooks, allowing for a more seamless transition from exploratory analysis to manuscript-ready figures. We could also explore integrating automated tools for checking style, grammar, and scientific rigour, further enhancing the system's role as a comprehensive quality control platform. Ultimately, RXiv-Maker is a contribution to a more open, efficient, and reproducible future for scientific communication, providing a powerful and accessible tool for modern researchers.

MANUSCRIPT PREPARATION

This manuscript was prepared using RXiv-Maker version 1.11.0.

DATA AVAILABILITY

Arxiv monthly submission data used in this article is available at `https://arxiv.org/stats/monthly_submissions`. The source code and data for the figures in this article are available at `https://github.com/henriques/rxiv-maker`.

CODE AVAILABILITY

The RXiv-Maker computational framework is available at `https://github.com/henriques/rxiv-maker`. All source code is under an MIT License.

AUTHOR CONTRIBUTIONS

Both Bruno M. Saraiva, Guillaume Jacquemet and Ricardo Henriques conceived the project and designed the framework. All authors contributed to writing and reviewing the manuscript.

ACKNOWLEDGEMENTS

B.S. and R.H. acknowledge support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001332) (to R.H.) and funding from the European Union through the Horizon Europe program (AI4LIFE project with grant agreement 101057970-AI4LIFE and RT-SuperES project with grant agreement 101099654-RTSuperES to R.H.). Funded by the European Union. However, the views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work was also supported by a European Molecular Biology Organization (EMBO) installation grant (EMBO-2020-IG-4734 to R.H.), a Chan Zuckerberg Initiative Visual Proteomics Grant (vpi-0000000044 with `https://doi.org/10.37921/743590vtudfp` to R.H.) and a Chan Zuckerberg Initiative Essential Open Source Software for Science (EOSS6-0000000260). This study was supported by the Academy of Finland (no. 338537 to G.J.), the Sigrid Juselius Foundation (to G.J.), the Cancer Society of Finland (Syöpäjärjestöt, to G.J.) and the Solutions for Health strategic funding to Åbo Akademi University (to G.J.). This research was supported by InFLAMES Flagship Program of the Academy of Finland (decision no. 337531).

EXTENDED AUTHOR INFORMATION

- **Bruno M. Saraiva:** 0000-0002-9151-5477; Bruno_MSaraiva; bruno-saraiva
- **Guillaume Jaquemet:** 0000-0002-9286-920X; guijacquemet; guijacquemet.bsky.social
- **Ricardo Henriques:** 0000-0002-1234-5678; HenriquesLab; henriqueslab.bsky.social; ricardo-henriques

Bibliography

1. Jeremy M Berg, Narinder Bhalla, Philip E Bourne, M Cacchione, D Egnor, J Flombaum, E Ghanem, M Ginsburg, B Goldstein, L Gorodetski, et al. Preprint servers. *Science*, 352 (6288):899, 2016. doi: 10.1126/science.aaf9133.
2. Richard J Abdill and Ran Blekhman. The growth of biorxiv preprints and the implications for preprint discovery. *PLoS Biology*, 17(4):e3000269, 2019. doi: 10.1371/journal.pbio.3000269.
3. Nicholas Fraser, Fakhri Momeni, Philipp Mayr, and Isabella Peters. The relationship between biorxiv preprints, citations and altmetrics. *Quantitative Science Studies*, 2(2):618–638, 2021. doi: 10.1162/qss_a_00043.
4. Ronald D Vale. Accelerating scientific publication in biology. *Proceedings of the National Academy of Sciences*, 116(52):26222–26229, 2019. doi: 10.1073/pnas.1915668116.
5. Jonathan P Tennant, François Waldner, Damien C Jacques, Paola Masuzzo, Lauren B Collister, and Chris HJ Hartgerink. The academic, economic and societal impacts of open access: an evidence-based review. *F1000Research*, 5:632, 2016. doi: 10.12688/f1000research.8460.3.
6. Vincent Lariviere, Cassidy R Sugimoto, Benoît Macaluso, Staša Milojević, and Blaise Cronin. arxiv. org and the dissemination of scholarly information: The case of computer science. *Journal of the Association for Information Science and Technology*, 65(4):844–848, 2014. doi: 10.1002/asi.23081.
7. Jerson L da Silva. The challenge of being a scientist in the 21st century. *Anais da Academia Brasileira de Ciências*, 94, 2022. doi: 10.1590/0001-3765202220211396.
8. David L Donoho. An invitation to reproducible computational research. *The American Statistician*, 64(1):1–2, 2010. doi: 10.1198/tast.2010.09132.

9. Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10):e1003285, 2013. doi: 10.1371/journal.pcbi.1003285.
10. Greg Wilson, Dhavide A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. Best practices for scientific computing. *PLoS Biology*, 12(1):e1001745, 2014. doi: 10.1371/journal.pbio.1001745.
11. Dominik Scherer, Lars Bendix, Stephan Greiner, and Julian Lück. A survey on the state of research data reproducibility in computer science. In *2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW)*, pages 310–317. IEEE, 2020. doi: 10.1109/ICSEW50701.2020.00052.
12. John D Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
13. Michael L Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021.
14. Mermaid Team. Mermaid: Generation of diagrams and flowcharts from text in a similar manner as markdown, 2023. Accessed: 2024-12-01.
15. Ricardo Henriques. Henriques biorxiv template, 2015. Overleaf LaTeX template. Accessed: 2025-06-16.
16. Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015. doi: 10.1145/2723872.2723882.
17. Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016. doi: 10.3233/978-1-61499-649-1-87.
18. Donald E Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984. doi: 10.1093/comjnl/27.2.97.
19. Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, 8(1):7, 2013. doi: 10.1186/1751-0473-8-7.
20. Yasset Perez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J Eglen, Daniel S Katz, et al. Ten simple rules for taking advantage of git and github. *PLoS Computational Biology*, 12(7):e1004947, 2016. doi: 10.1371/journal.pcbi.1004947.
21. Ivan Hidalgo-Cenalmor, Joanna W Pylvänäinen, Mariana G Ferreira, Craig T Russell, Alon Saguy, Ignacio Arganda-Carreras, Guillaume Jacquemet, and Ricardo Henriques. Dl4miceverywhere: deep learning for microscopy made flexible, shareable and reproducible. *Nature Methods*, 21(5):804–810, 2024. doi: 10.1038/s41592-024-02295-6.
22. John MacFarlane. Pandoc: A universal document converter. *Journal of Open Source Software*, 7(76):4642, 2022. doi: 10.21105/joss.04642.
23. Jamie J Kirkham, Christopher M Penfold, Fiona Murphy, and John PA Ioannidis. The quality of reporting in preprints. *bioRxiv*, page 451822, 2018. doi: 10.1101/451822.

Methods

The RXiv-Maker system is architected as a modular pipeline orchestrated by a central Makefile, which automates all stages of manuscript production from markdown source to a final PDF document. The core conversion from markdown to LaTeX is handled by custom scripts, leveraging the versatility of Python to accommodate specialised scientific formatting requirements not natively supported in standard markdown. For bibliographic management, the system integrates seamlessly with BibTeX, utilising .bib files to manage references. These are automatically processed to generate formatted citations and a complete bibliography within the final document. The project structure is standardised to ensure clarity and ease of use. All manuscript text is located in markdown files within a designated src/ directory. Figures can be either static images placed in src/figures/ or generated programmatically by scripts in the scripts/ directory. A YAML file, metadata/metadata.yaml, centrally stores all manuscript metadata, such as title, authors, affiliations, and abstract, separating content from presentation. This separation is critical for programmatic access and modification of document properties without altering the narrative text.

Compilation can be initiated through several mechanisms. The make command compiles the document locally, provided the necessary dependencies like a LaTeX distribution are installed. To eliminate the need for manual installation of dependencies, we provide a Dockerfile that defines the complete software environment. Building this Dockerfile creates a container image from which a user can compile the manuscript with a single Docker command, ensuring perfect reproducibility of the environment. The build script (build.sh) automates the creation of this Docker image, supporting multi-architecture builds for both amd64 and arm64 platforms to ensure broad compatibility. For continuous integration and cloud-based compilation, a GitHub Actions workflow file, .github/workflows/main.yml, is included. This workflow automates the entire process of building the Docker container, compiling the manuscript, and releasing the resulting PDF as a downloadable artefact on every push to the repository, thereby providing a robust, automated pipeline for manuscript generation. This automated workflow serves as a powerful tool for quality control and immediate dissemination.

Supplementary Information

Rχiv-Maker: An Automated Template Engine for Streamlined Scientific Publications

Generated on June 22, 2025 by RXiv-Maker

Markdown Element	LaTeX Equivalent	Description
`**bold text**`	`\{}textbf{bold text}`	`Bold formatting for emphasis`
`*italic text*`	`\{}textit{italic text}`	`Italic formatting for emphasis`
`# Header 1`	`\{}section{Header 1}`	`Top-level section heading`
`## Header 2`	`\{}subsection{Header 2}`	`Second-level section heading`
`### Header 3`	`\{}subsubsection{Header 3}`	`Third-level section heading`
`citation`	`\{}cite{citation}`	`Single citation reference`
`[cite1;cite2]`	`\{}cite{cite1,cite2}`	`Multiple citation references`
`fig:label`	`\{}ref{fig:label}`	`Figure cross-reference`
`Image with attributes`	`\{}begin{figure}...\{}end{figure}`	`Figure with attributes (old format)`
`Image with caption`	`\{}begin{figure}...\{}end{figure}`	`Figure with separate caption (new format)`
`- list item`	`\{}begin{itemize}\{}item...\{}end{itemize}`	`Unordered list`
`1.  list item`	`\{}begin{enumerate}\{}item...\{}end{enumerate}`	`Ordered list`
`[link text](url)`	`\{}href{url}{link text}`	`Hyperlink with custom text`
`https://example.com`	`\{}url{https://example.com}`	`Bare URL`
`<!- comment ->`	`% comment`	`Comments (converted to LaTeX style)`
`Markdown table`	`\{}begin{table}...\{}end{table}`	`Table with automatic formatting`
`<newpage>`	`\{}newpage`	`Manual page break control`
`<clearpage>`	`\{}clearpage`	`Page break with float clearing`

Sup. Table S1. RXiv-Maker Markdown Syntax Overview. Comprehensive mapping of markdown elements to their LaTeX equivalents, demonstrating the automated translation system that enables researchers to write in familiar markdown syntax whilst producing professional LaTeX output.

Deployment Method	Environment	Dependencies	Collaboration	Ease of Use	Reproducibility
Docker Local	Local machine	Docker only	Git-based	High	Perfect
GitHub Actions	Cloud CI/CD	None (cloud)	Automatic	Very High	Perfect
Google Colab	Web browser	None (cloud)	Shared notebooks	Very High	High
Local Python	Local machine	Python + LaTeX	Git-based	Medium	Good
Manual LaTeX	Local machine	Full LaTeX suite	Git-based	Low	Variable

Sup. Table S2. RXiv-Maker Deployment Strategies. Comparison of available compilation methods, highlighting the flexibility of the framework in accommodating different user preferences and technical environments whilst maintaining consistent output quality.

Directory	Purpose	Content Types	Version Control	Processing Stage
`MANUSCRIPT/`	Scientific content	Markdown, YAML, BibTeX	Full tracking	Source
`FIGURES/`	Visual content	Python scripts, Mermaid, data	Full tracking	Source + Generated
`src/`	Framework code	Python modules, templates	Full tracking	Processing
`output/`	Compilation workspace	LaTeX, PDF, auxiliaries	Excluded (.gitignore)	Output
`build/`	Docker environment	Container definitions	Full tracking	Infrastructure

Sup. Table S3. Project Organisation Schema. Systematic arrangement of project components that facilitates clear separation of concerns, enhances maintainability, and supports collaborative development workflows whilst ensuring computational reproducibility.

Supp. Note 1: Architectural Philosophy and Project Organisation.. The RXiv-Maker framework embodies a carefully considered architectural philosophy that prioritises clarity, maintainability, and computational reproducibility through systematic organisation of project components. The system's file structure reflects established software engineering principles whilst

Format	Input Extension	Processing Method	Output Formats	Quality	Use Case
Mermaid Diagrams	`.mmd`	Mermaid CLI	SVG, PNG, PDF	Vector/Raster	Flowcharts, architectures
Python Figures	`.py`	Script execution	PNG, PDF, SVG	Publication	Data visualisation
Static Images	`.png`, `.jpg`, `.svg`	Direct inclusion	Same format	Original	Photographs, logos
LaTeX Graphics	`.tex`, `.tikz`	LaTeX compilation	PDF	Vector	Mathematical diagrams
Data Files	`.csv`, `.json`, `.xlsx`	Python processing	Via scripts	Computed	Raw data integration

Sup. Table S4. Supported Figure Generation Methods. Comprehensive overview of the framework's figure processing capabilities, demonstrating support for both static and dynamic content generation with emphasis on reproducible computational graphics.

accommodating the specific requirements of scientific manuscript preparation. This organisational schema segregates content, configuration, and computational elements into distinct hierarchical domains, thereby facilitating both human comprehension and automated processing.

The primary manuscript content resides within the MANUSCRIPT directory, which houses the core intellectual contribution in easily accessible formats. This directory contains the YAML configuration file (00_CONFIG.yml) that centralises all metadata including authorship details, institutional affiliations, and document properties, thereby enabling programmatic manipulation of manuscript attributes without requiring modifications to the narrative content. The numbered markdown files (01_MAIN.md, 02_SUPPLEMENTARY_INFO.md) contain the substantive text, with the numerical prefixing ensuring logical processing order whilst maintaining intuitive organisation for collaborative authoring. The BibTeX references file (03_REFERENCES.bib) provides standardised bibliographic management, ensuring consistent citation formatting across the entire document. Figure sources and data are organised within dedicated subdirectories (FIGURES/, TABLES/) that maintain clear separation between content types whilst enabling automated discovery during the compilation process.

The src directory encompasses the computational infrastructure that transforms markdown source into publication-ready output. This separation ensures that the technical implementation remains distinct from the scientific content, facilitating maintenance and updates to the processing pipeline without affecting the manuscript itself. The modular structure within src reflects software engineering best practices, with specialised processors for different content types that can be independently developed and tested. The output directory serves as the compilation workspace where intermediate files and final products are generated, preventing contamination of source materials with temporary compilation artefacts whilst providing transparency into the conversion process.

Supp. Note 2: Comparative Analysis with Alternative Scientific Authoring Platforms.. Within the broader landscape of scientific authoring tools, RXiv-Maker occupies a distinctive position that reflects careful consideration of the trade-offs between functionality and simplicity. Platforms such as Overleaf have revolutionised collaborative LaTeX authoring by providing sophisticated web-based environments with real-time collaboration features, comprehensive template libraries, and integrated compilation services. These systems excel in scenarios requiring complex document structures, advanced typesetting control, and seamless multi-author workflows. The platform's strength lies in its ability to democratise LaTeX authoring by providing a familiar word-processor-like interface whilst maintaining the typographical excellence of LaTeX output.

Similarly, Quarto represents a powerful framework for scientific and technical publishing that supports multiple programming languages, diverse output formats, and sophisticated computational document features. Its versatility enables researchers to create documents that seamlessly integrate narrative text with executable code, supporting formats ranging from HTML web pages to PDF documents and interactive presentations. Quarto's strength lies in its comprehensive approach to scientific communication, enabling complex multi-format publishing workflows across various scientific domains.

Pandoc, as a universal document converter, provides exceptional flexibility in transforming content between numerous formats. Its strength lies in its ability to serve as a foundation for custom publishing workflows, enabling researchers to develop bespoke solutions for specific requirements. However, this flexibility comes at the cost of increased complexity in configuration and setup.

RXiv-Maker deliberately positions itself as a complementary tool that prioritises simplicity and focused functionality over comprehensive feature coverage. Whilst acknowledging the considerable strengths of these established platforms, RXiv-Maker addresses a specific niche within the scientific publishing ecosystem: the efficient production of high-quality preprints for repositories such as arXiv, bioRxiv, and medRxiv. This focused approach enables optimisation for this particular use case, resulting in a streamlined workflow that minimises cognitive overhead for researchers primarily concerned with rapid dissemination of their findings. The framework's emphasis on markdown as the primary authoring language reflects a philosophical commitment to accessibility and sustainability, providing an intuitive syntax that most researchers can master quickly whilst maintaining typographical excellence.

Supp. Note 3: Programmatic Figure Generation and Computational Reproducibility.. The technical architecture underlying RXiv-Maker's figure generation capabilities demonstrates how automated processing pipelines can maintain transparent connections between source data and final visualisations whilst ensuring computational reproducibility. The system

supports two primary methodologies for figure creation: Mermaid diagram processing and Python-based data visualisation, each addressing distinct requirements within the scientific publishing workflow.
Mermaid diagram processing leverages the Mermaid CLI to convert text-based diagram specifications into publication-ready graphics. This approach enables version-controlled diagram creation where complex flowcharts, system architectures, and conceptual models can be specified using intuitive syntax and automatically rendered into multiple output formats. The system generates SVG, PNG, and PDF variants to accommodate different compilation requirements whilst maintaining vector quality where appropriate. This automation eliminates the manual effort traditionally required for diagram creation and updates, whilst ensuring that modifications to diagram specifications are immediately reflected in the final document.
Python figure generation represents a more sophisticated approach to computational reproducibility, where analytical scripts are executed during document compilation to generate figures directly from source data. This integration ensures that visualisations remain synchronised with the underlying datasets and analytical methods, eliminating the possibility of outdated or inconsistent graphics persisting in the manuscript. The system executes Python scripts within the compilation environment, automatically detecting generated image files and incorporating them into the document structure. This approach transforms figures from static illustrations into dynamic, reproducible computational artefacts that enhance the scientific rigour of the publication.

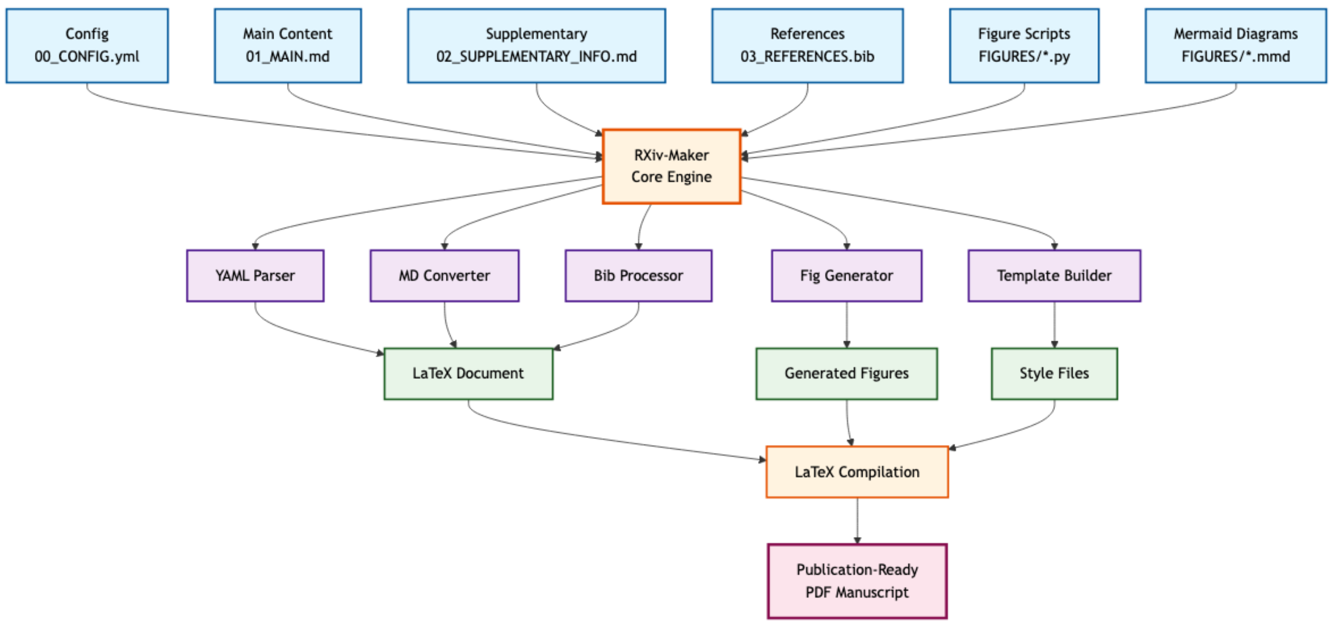
Supp. Note 4: Markdown-to-LaTeX Conversion Architecture and Processing Pipeline.. The markdown-to-LaTeX conversion architecture demonstrates how specialised processors can handle complex document transformations whilst maintaining code modularity and testability. The system employs dedicated processors for figures, tables, citations, and other content types, each implementing specific transformation rules that preserve semantic meaning whilst ensuring typographical excellence. This modular approach enables independent development and testing of conversion components, facilitating maintenance and enhancement of the framework's capabilities.
Figure processing supports multiple syntax variants to accommodate different authoring preferences, including the new format where images are followed by attribute blocks and captions, the attributed format with inline specifications, and simple format for basic inclusions. The core conversion function implements a multi-pass approach that protects literal content during transformation, processes each figure format through dedicated functions, and restores protected content after processing. This sophisticated content protection mechanism ensures that code examples and other literal content are preserved during transformation, proving essential for technical manuscripts.
Table processing handles GitHub Flavored Markdown tables with LaTeX-specific enhancements such as rotation capabilities and sophisticated cross-referencing systems. The conversion system supports both legacy and modern caption formats, enabling authors to specify table properties including width detection for double-column layouts, rotation angles for landscape orientation, and identifier extraction for cross-referencing. The table cell formatting function implements context-aware processing that preserves markdown syntax within examples whilst properly escaping special characters and converting emphasis markers to appropriate LaTeX commands.
Reference processing demonstrates how automated systems can enhance document quality whilst reducing authoring burden. The framework automatically converts markdown-style references into appropriate LaTeX cross-references, ensuring consistent formatting and enabling LaTeX's sophisticated reference management capabilities. This automation extends to bibliographic citations, where the system integrates seamlessly with BibTeX workflows to provide professional citation formatting without requiring authors to master LaTeX citation syntax.

Supp. Note 5: Reproducibility Features and Version Control Integration.. The RXiv-Maker framework incorporates reproducibility as a fundamental design principle rather than an afterthought, implementing features that ensure complete traceability from source data to final publication. The system's integration with Git version control provides comprehensive tracking of all components necessary for manuscript generation, including content files, configuration parameters, processing scripts, and even the framework code itself. This approach ensures that every aspect of the publication process can be reproduced, verified, and audited.
The containerised compilation environment, implemented through Docker, provides perfect isolation and reproducibility of the software environment. By encapsulating the exact versions of LaTeX, Python libraries, and system dependencies within a container image, the framework eliminates the common "works on my machine" problem that plagues many scientific computing workflows. This containerisation extends beyond mere convenience to serve as a critical component of scientific integrity, ensuring that the same input always produces identical output regardless of the host system configuration.
The framework's programmatic approach to figure generation creates an auditable chain from raw data to final visualisation. Python scripts that generate figures are version-controlled alongside the manuscript content, enabling complete reconstruction of all visual elements from source data. This approach contrasts sharply with traditional workflows where figures are created separately and inserted as static images, potentially leading to inconsistencies when data is updated or analysis methods are refined.

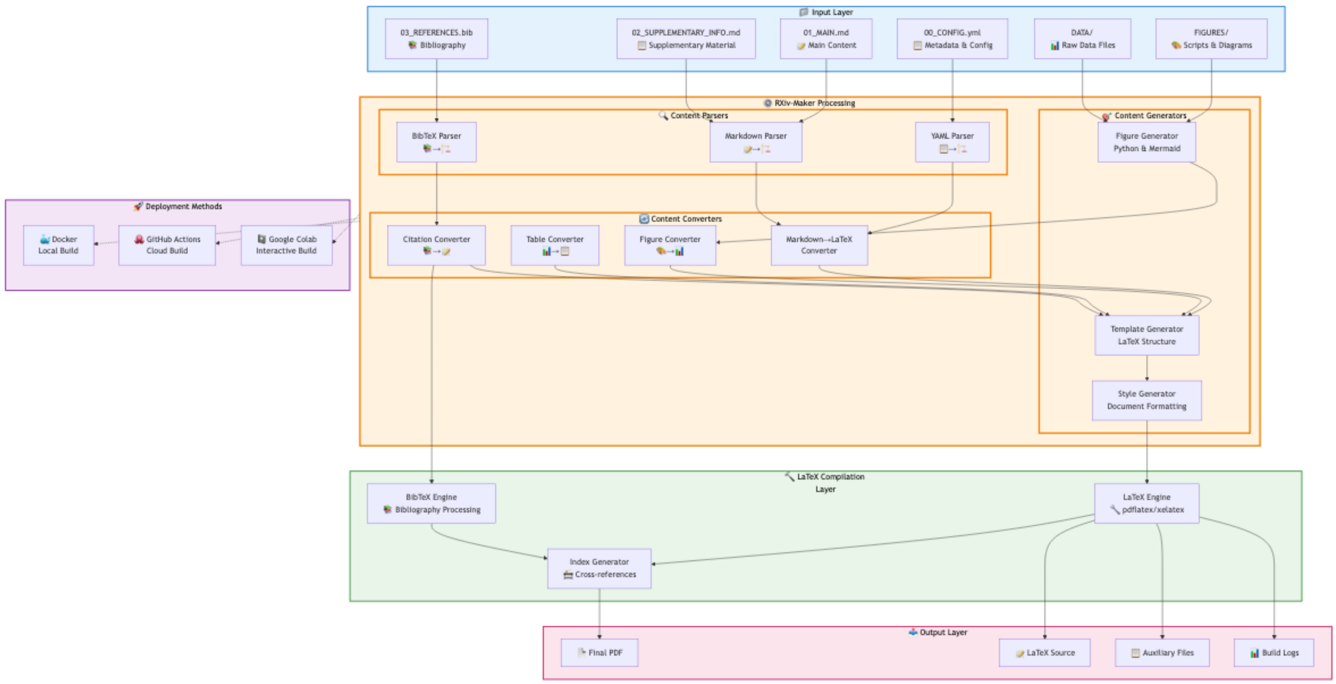
Supp. Note 6: Template Customisation and Advanced Styling Options.. The RXiv-Maker framework provides extensive customisation capabilities through its LaTeX template system, enabling researchers to adapt the visual presentation to meet specific publication requirements whilst maintaining the simplicity of the markdown authoring experience. The template architecture separates content from presentation through a sophisticated class file (rxiv_maker_style.cls) that encapsulates all formatting decisions, typography choices, and layout specifications.

The YAML configuration system enables fine-grained control over document properties including author information formatting, institutional affiliation handling, and abstract presentation. Advanced users can modify template parameters to adjust margins, typography, colour schemes, and sectioning styles without requiring direct LaTeX modifications. The framework supports customisation of citation styles through configurable BibTeX style files, enabling compliance with specific journal requirements or institutional guidelines.

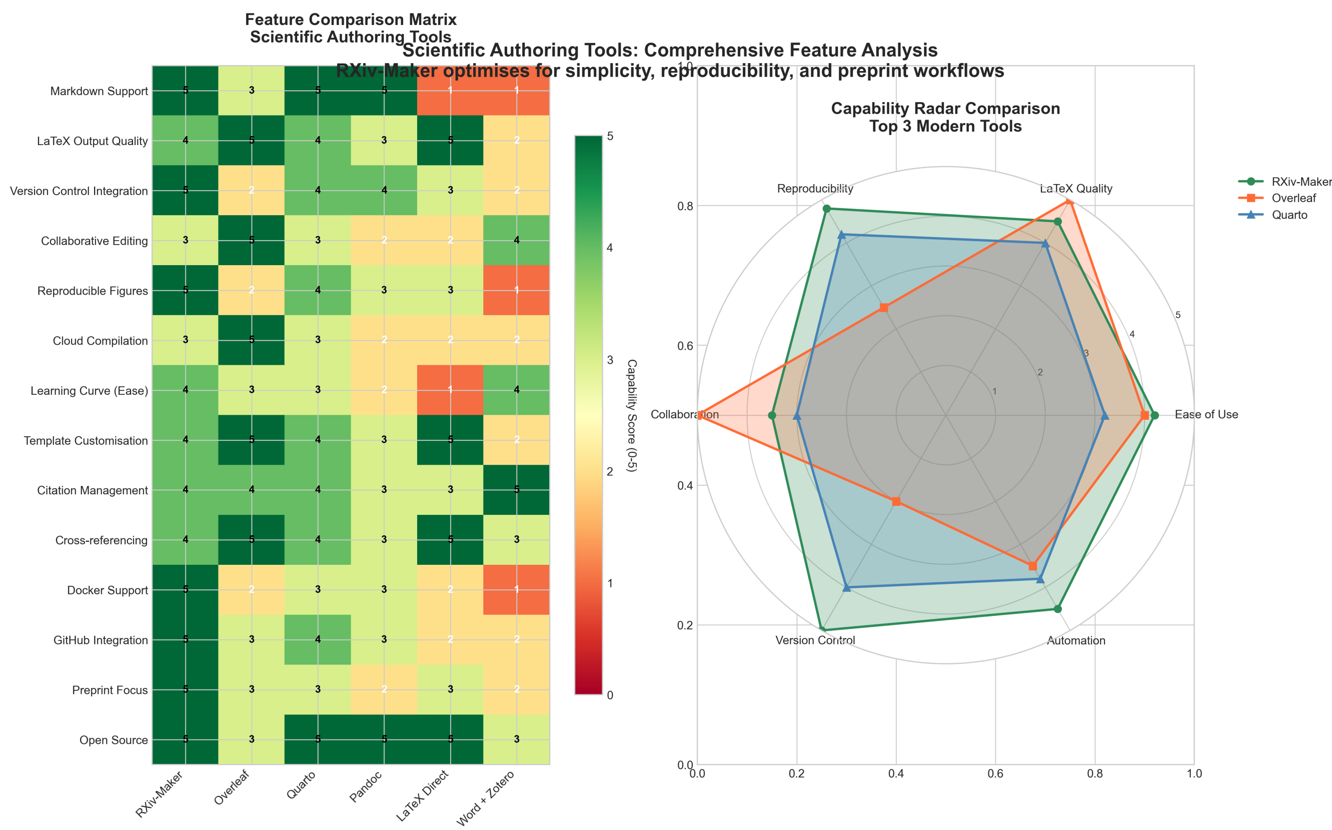
For institutions requiring consistent branding or specific formatting requirements, the framework provides extension points that enable custom style development whilst maintaining compatibility with the core processing pipeline. This extensibility ensures that RXiv-Maker can adapt to diverse institutional requirements without compromising its fundamental commitment to simplicity and ease of use.



Sup. Fig. S1. RXiv-Maker Workflow Overview. Simplified representation of the RXiv-Maker system architecture, illustrating how the standardised file naming convention (00_CONFIG.yml, 01_MAIN.md, 02_SUPPLEMENTARY_INFO.md, 03_REFERENCES.bib) integrates with the processing engine to generate publication-ready documents through a fully automated pipeline from markdown input to PDF output.



Sup. Fig. S2. Detailed System Architecture and Processing Layers. Comprehensive technical diagram showing the complete RXiv-Maker architecture, including input layer organisation, processing engine components (parsers, converters, generators), compilation infrastructure, output generation, and deployment methodology integration. This figure illustrates the modular design that enables independent development and testing of system components.



Sup. Fig. S3. Feature Comparison Analysis Across Scientific Authoring Platforms. Quantitative comparison of RXiv-Maker capabilities relative to established scientific authoring tools including Overleaf, Quarto, Pandoc, and traditional LaTeX workflows. The heatmap demonstrates relative strengths across key functionality areas, whilst the radar chart provides detailed capability analysis for modern tools, highlighting RXiv-Maker's optimisation for simplicity, reproducibility, and preprint-focused workflows.