Rχiv-Forge: 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 shifted towards rapid dissemination through preprint servers, placing increased demands on researchers for manuscript preparation and quality control. We present RXiv-Forge, a comprehensive GitHub-native system that simplifies scientific writing through markdown-based authoring with automated LaTeX conversion. The system enables researchers to write in familiar markdown syntax whilst producing publication-quality documents with professional typesetting. RXiv-Forge offers flexible compilation strategies ranging from GitHub Actions and Google Colab to local compilation with minimal installation requirements using Docker containerisation. The framework supports diverse visualisation approaches, including programmatic figure generation through Python libraries such as Matplotlib and Seaborn, alongside integrated Mermaid diagram rendering for conceptual illustrations. This self-documenting article demonstrates the system's capabilities, showcasing how markdown-centric authoring transforms scientific communication into an efficient, collaborative, and reproducible process. RXiv-Forge serves as a foundational tool for research groups seeking streamlined publishing workflows, enabling scientists to focus on research content whilst maintaining rigorous technical standards.

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*

Fig. 1. The RXiv-Forge 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.

Main

Scientific publishing has undergone profound transformation over the past two decades, fundamentally altering how researchers communicate and disseminate findings (1). The emergence of preprint servers such as arXiv, bioRxiv, and medRxiv has enabled millions of manuscripts to bypass traditional peer-review bottlenecks, with exponential growth particularly evident during the COVID-19 pandemic (2, 3). This shift towards immediate publication reflects growing recognition that traditional publishing timelines are incompatible with the pace of modern scientific discovery.

Concurrent with this transformation, computational tools and automated workflows have become essential to contemporary research practice. Version control systems, particularly Git and GitHub, have evolved into platforms for scientific collaboration, enabling transparent manuscript development and reproducible analyses (4, 5). Containerisation technologies such as Docker have enhanced reproducibility by providing standardised computational environments (6, 7). Traditional manuscript p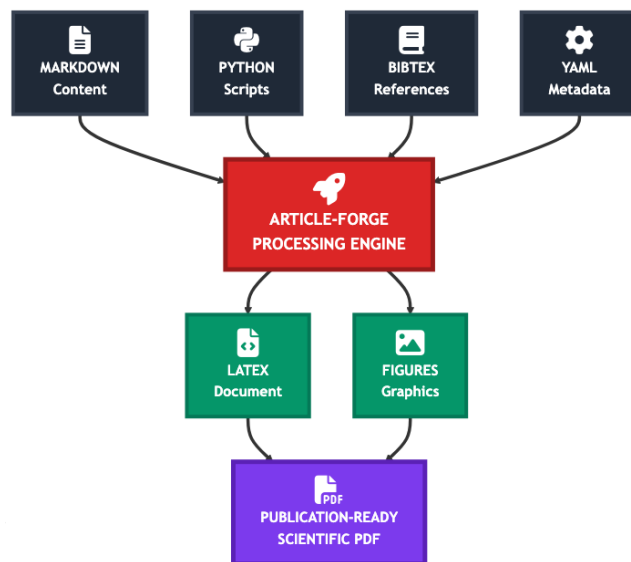reparation remains largely unchanged, relying on fragmented workflows that separate content creation, figure generation, and document compilation. This fragmentation introduces opportunities for error and version conflicts. Contemporary research demands sophisticated figure generation capabilities integrating statistical analysis and publication-quality visualisation. Whilst Python libraries such as Matplotlib and Seaborn provide foundational tools for scientific visualisation (8, 9), modern workflows also benefit from declarative diagram creation through Mermaid, enabling researchers to generate flowcharts and conceptual illustrations using text-based syntax.

RXiv-Forge 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 (10), 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. 1, provides automated figure generation for statistical visualisation, integrated Mermaid diagram creation, and robust build automation through containerised environments. RXiv-Forge provides extensive capabilities for programmatic
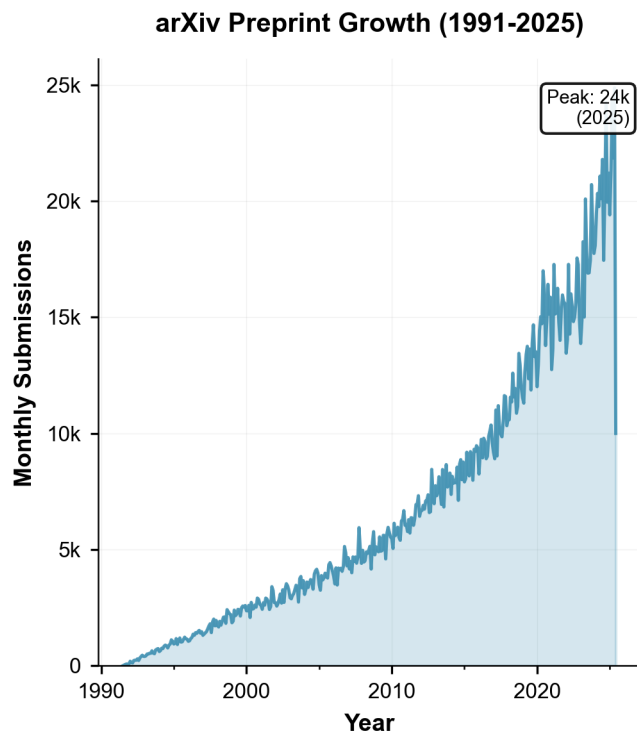
arXiv Preprint Growth (1991-2025)

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-Forge pipeline. This demonstrates the system's capacity for reproducible, data-driven figure generation directly within the publication workflow.

and reproducible figure generation directly from underlying data and source code, though researchers may choose traditional static figure inclusion based on their workflow preferences. When employed, this approach ensures that visualisations become dynamic artefacts, intrinsically linked to the research process and subject to the same rigorous version control as the manuscript text itself.

This figure is rendered automatically during the article's compilation by executing a version-controlled Python script (`Figure_2.py`) found in `FIGURES/`. The script uses the Matplotlib and Pandas libraries to process a dataset of arxiv monthly submission statistics, whose snapshot copy is maintained within the repository. This methodology exemplifies a core aspect of transparent and reproducible science: the unbreakable link between data, analysis, and the resulting visualisation. Any modification to the dataset or the visualisation code will be automatically reflected in the manuscript upon recompilation, thus ensuring complete transparency, eliminating the possibility of data-figure mismatch, and allowing for full verifiability by peers. This self-generating figure serves as a direct validation of the RXiv-Forge system's capacity to streamline and safeguard the integrity of scientific reporting.

Core Functionality Overview

RXiv-Forge provides comprehensive manuscript authoring capabilities through its integrated toolchain. The system supports multiple input formats including standard markdown syntax, YAML-based metadata configuration, and BibTeX

bibliography management. Figure generation encompasses both programmatic approaches through Python scripts and declarative diagram creation using Mermaid syntax, enabling researchers to maintain complete reproducibility whilst accommodating diverse visualisation requirements.

The compilation pipeline offers multiple deployment strategies to accommodate different research environments and computational preferences. Users can leverage GitHub Actions for automated cloud-based compilation, utilise Google Colab for interactive development and testing, or deploy locally using Docker containers with minimal system requirements. This flexibility ensures that RXiv-Forge remains accessible across different computational environments and institutional constraints.

Comparative Analysis

Table* 1: presents a detailed comparison of RXiv-Forge against established manuscript preparation systems, highlighting key differentiators in workflow integration, reproducibility features, and deployment flexibility.

Feature	RXiv-Forge	Ov
Input Format	Markdown + YAML	LaT
Learning Curve	Low	Me
Version Control	Native Git	Lir
Reproducible Figures	Python + Mermaid	Ma
Collaboration	GitHub-native	We
Deployment Options	GitHub Actions, Colab, Local Docker	We
Template System	Automated conversion	Ma
Citation Management	BibTeX integration	Bib
Containerisation	Docker-native	N/
Continuous Integration	GitHub Actions	N/

RXiv-Forge responds directly to evolving demands of modern scientific communication. The programmatic generation of Figure 2 within this document validates our framework, demonstrating how figures become reproducible and verifiable components of the scientific record rather than static images. This approach mitigates common errors whilst the integration of Git, Docker, and GitHub Actions establishes a research environment where transparency and collaboration are structurally embedded. RXiv-Forge provides a foundational tool for research groups adopting structured, automated publishing approaches, enabling scientists to focus on research content whilst ensuring efficient and robust dissemination processes.

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. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015. doi: 10.1145/2723872.2723882.
7. 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.
8. John D Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
9. Michael L Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021.
10. Ricardo Henriques. Henriqueslab biorxiv template, 2015. Overleaf LaTeX template. Accessed: 2025-06-16.

Methods

The RXiv-Forge framework orchestrates a series of computational tools to achieve a fully automated publication pipeline. The process begins with manuscript content authored in Markdown (01_MAIN.md) and metadata defined in a separate YAML configuration file (00_CONFIG.yml). Bibliographic information is managed in a standard BibTeX file (03_REFERENCES.bib). The core of the system is a set of Python scripts located in src/py/ which parse the Markdown and YAML to dynamically generate a main LaTeX file (MANUSCRIPT.tex) from a template (src/tex/template.tex). Figure generation is a key automated step. Mermaid diagrams (.mmd) and Python scripts (.py) placed in the FIGURES/ directory are executed to produce visual content. For instance, Figure 2 was generated by executing `FIGURES/Figure_2.py`, which processes data from `FIGURES/DATA/Figure_2/arxiv_monthly_submissions.csv`. The entire build process is managed by a Makefile and can be encapsulated within a Docker container defined by the Dockerfile, ensuring a consistent and reproducible compilation environment. Continuous integration and deployment are handled by GitHub Actions, which automates the compilation of the PDF upon every commit, making the latest version of the manuscript perpetually available.

Note 1: Supplementary Information

RXiv-Forge: An Automated Template Engine for Streamlined Scientific Publications

This supplementary information provides additional technical details, implementation examples, and extended documentation for the RXiv-Forge framework. The content is organised into supplementary figures that illustrate system architecture and functionality, followed by supplementary notes that detail technical implementation aspects and provide practical examples of the auto-translation system.
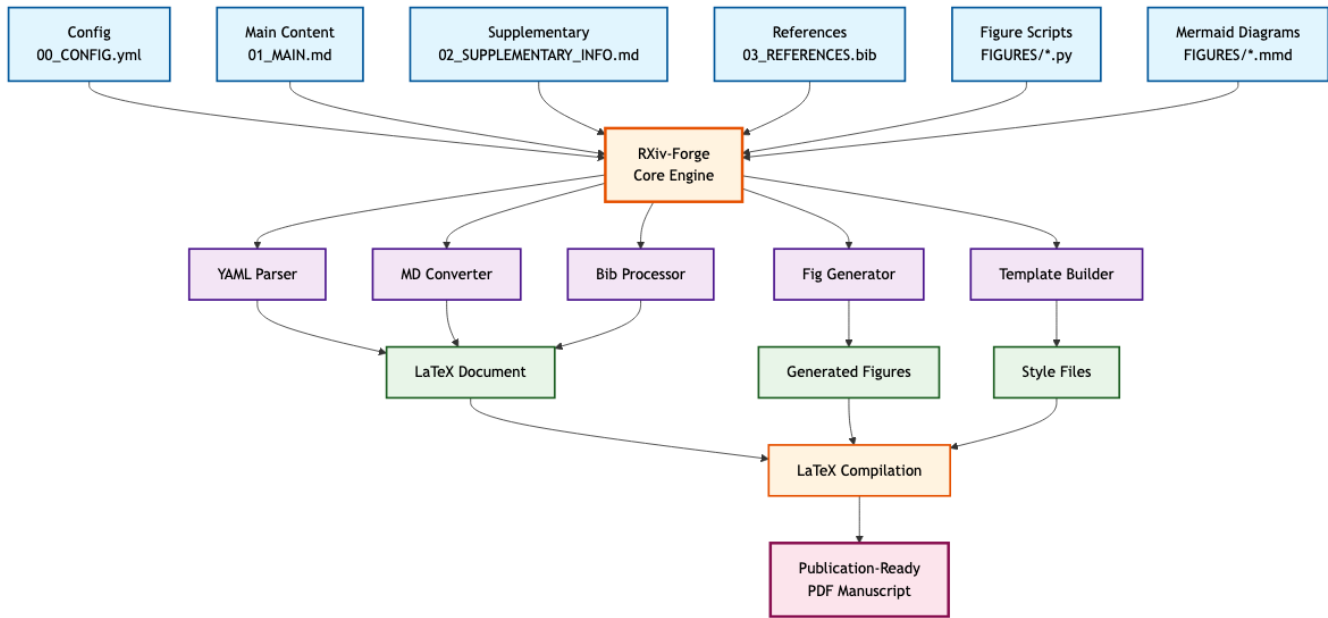
Sup. Fig. 1. RXiv-Forge Workflow Details. This figure provides a comprehensive overview of the RXiv-Forge system architecture, showing how the simplified file naming convention (00_CONFIG.yml, 01_MAIN.md, 02_SUPPLEMENTARY_INFO.md, 03_REFERENCES.bib) integrates with the processing engine to generate publication-ready documents. The system demonstrates the complete automation pipeline from markdown input to PDF output.

A. Supplementary Figures.

B. Supplementary Tables.

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)
Fenced code block	`\{}begin{verbatim}...\{}end{verbatim}`	Multi-line code block
`- 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

Sup. Table 1. RXiv-Forge Markdown Syntax Overview. Comprehensive overview of RXiv-Forge's markdown to LaTeX conversion capabilities, demonstrating the automated translation system that enables researchers to write in familiar markdown syntax while producing professional LaTeX output.

B.1. RXiv-Forge Markdown Syntax Overview.

C. Supplementary Notes.

C.1. File Structure and Organisation. Blablabla

C.2. Figures and tables. Blablabla

C.3. Comparison with similar systems.

C.4. Supplementary Note 2: Auto-Translation System Examples. The RXiv-Forge auto-translation system processes structured input files to generate professional LaTeX output. The following examples demonstrate the system's capabilities across different file types.

YAML Configuration Example (00_CONFIG.yml)

```
title: "RXiv-Forge: An Automated Template Engine for Streamlined Scientific Publications"
short_title: "RXiv-Forge"
authors:
\begin{itemize}
  \item name: "Bruno M. Saraiva"
\end{itemize}
    affiliation: [1, 2]
    email: "bruno.saraiva\cite{example}.com"
    orcid: "0000-0000-0000-0000"
\begin{itemize}
  \item name: "Guillaume Jaquemet"
\end{itemize}
    affiliation: [3]
    email: "guillaume.jaquemet\cite{example}.com"
    orcid: "0000-0000-0000-0000"
\begin{itemize}
  \item name: "Ricardo Henriques"
\end{itemize}
    affiliation: [1, 2]
    email: "ricardo.henriques\cite{example}.com"
    orcid: "0000-0000-0000-0000"
    corresponding: true

affiliations:
  1: "Instituto Gulbenkian de Ciência, Oeiras, Portugal"
  2: "University College London, London, United Kingdom"
  3: "Åbo Akademi University, Turku, Finland"

abstract: "Modern scientific publishing requires..."
keywords: ["scientific publishing", "reproducibility", "automation"]
```

Markdown Content Structure (01_MAIN.md)

```
\subsection{Abstract}
Modern scientific publishing has shifted towards rapid dissemination...

\subsection{Main}
Scientific publishing has undergone profound transformation...

\begin{sfigure}[ht]
\centering
\includegraphics[width=\linewidth]{Figures/Figure\_1.png}
\caption{Figure caption with cross-reference}
\label{fig:1}
\end{sfigure}
\newpage
```

```
Statistical analysis demonstrates significant improvements \cite{reference2023}.

\subsection{Methods}
The RXiv-Forge framework orchestrates computational tools...
```

BibTeX Reference Format (03_REFERENCES.bib)

```
\cite{article}{Tennant2016_academic_publishing,
  title={The academic, economic and societal impacts of Open Access},
  author={Tennant, Jonathan P and Waldner, Fran{\c{c}}ois and Jacques, Damien C},
  journal={PLoS Biology},
  volume={14},
  number={7},
  pages={e1002510},
  year={2016},
  publisher={Public Library of Science}
}

\cite{article}{Fraser2021_preprint_growth,
  title={The relationship between bioRxiv preprints and citations},
  author={Fraser, Nicholas and Momeni, Fakhri and Mayr, Philipp and Peters, Isabella},
  journal={Quantitative Science Studies},
  volume={2},
  number={2},
  pages={618--638},
  year={2021}
}
```

C.5. Supplementary Note 3: Technical Implementation Pipeline. The system processes these files through a sophisticated conversion pipeline:

1. **Configuration Parsing**: Extracts metadata from YAML configuration file, including author information, affiliations, and document settings

2. **Content Conversion**: Transforms markdown syntax into LaTeX formatting, preserving cross-references, citations, and figure placements

3. **Figure Generation**: Executes Python scripts and processes Mermaid diagrams automatically during compilation

4. **Document Assembly**: Combines all components into a cohesive LaTeX document using the template system

5. **Citation Processing**: Integrates BibTeX references with proper formatting and cross-referencing

6. **Output Compilation**: Produces publication-ready PDF with professional typesetting and formatting

This approach ensures reproducibility, version control compatibility, and automated processing whilst maintaining the flexibility needed for academic publishing. The system automatically handles complex LaTeX formatting requirements, enabling researchers to focus on content creation rather than technical implementation details.