

Supplementary Material

First Author^a, Second Author^b, Corresponding Author^{a,*}

^a*First Institution, Department, City, Country*

^b*Second Institution, Department, City, Country*

~ 0 supplementary figures, 0 supplementary tables, 1077 words for supplementary text

1. Supplementary Methods

Supplementary Methods

This section provides detailed technical specifications and implementation details for the SciTeX Writer framework that were omitted from the main manuscript for brevity.

Container Image Construction

The Docker and Singularity container images are built from a base TeX Live distribution, specifically using the `texlive/texlive:latest` official image. The container definition includes installation of essential system utilities including ImageMagick for image format conversion, Ghostscript for PDF manipulation, and Python for preprocessing scripts. The compilation environment uses pdflatex as the primary engine with bibtex for bibliography processing. The container image size is approximately 3.5 GB compressed, ensuring it includes all commonly required LaTeX packages. Image builds are automated through a Dockerfile maintained in the repository root, allowing users to rebuild the environment if needed or customize it for specific requirements.

*Corresponding author. Email: corresponding.author@institution.edu

24 *Makefile Command Reference*

25 The Makefile provides a comprehensive set of targets for document compila-
26 tion and management. The `make manuscript` command compiles the main
27 manuscript by first executing preprocessing scripts, then running `pdflatex`
28 twice, followed by `bibtex`, and finally `pdflatex` twice more to resolve all cross-
29 references. The `make clean` target removes auxiliary files while preserving
30 source content and compiled PDFs. The `make archive` command creates a
31 timestamped copy of the current manuscript in the archive directory using
32 the format `manuscript_vXXX.tex` where XXX is an automatically incre-
33 mented version number. The `make diff` target executes `latexdiff` between
34 the current version and the most recent archived version, producing a PDF
35 with color-coded additions and deletions.

36 *Preprocessing Pipeline Implementation*

37 Figure preprocessing involves scanning the `01_manuscript/contents/figures/caption_and_m`
38 directory for subdirectories containing image files and corresponding `.tex`
39 caption files. The script extracts the caption text, determines the appropri-
40 ate image file based on priority ordering (PDF, then PNG, then JPEG), and
41 generates LaTeX figure inclusion code using the `graphicx` package. The gen-
42 erated code maintains aspect ratios, sets maximum widths to column width,
43 and includes proper labeling for cross-referencing. All generated figure code
44 is concatenated into `FINAL.tex` which is included by the main document.
45 The table preprocessing follows an analogous workflow but handles the ad-
46 ditional complexity of tabular environments and allows for both simple and
47 complex multi-column layouts.

48 *Version Control Integration*

49 The framework integrates with Git through hook scripts that can option-
50 ally be installed to trigger automatic archiving upon commit. The `.gitignore`
51 file is configured to exclude compilation artifacts including auxiliary files, log
52 files, and temporary directories while preserving source content, archived ver-
53 sions, and final PDFs. The repository structure is designed to minimize merge

54 conflicts by isolating frequently-modified content files from rarely-changed
55 configuration files. Branch-based workflows are supported, allowing authors
56 to develop different manuscript sections on feature branches before merging
57 to the main development branch.

58 *Cross-Reference Management*

59 The framework uses consistent labeling conventions for cross-references
60 throughout the document. Figures use the prefix `fig:`, tables use `tab:`,
61 sections use `sec:`, and equations use `eq:`. The preprocessing scripts au-
62 tomatically generate labels based on figure and table file names, ensuring
63 uniqueness without requiring manual label assignment. The `hyperref` pack-
64 age is configured to generate clickable links in the compiled PDF, with colors
65 customized to be visible in both digital and printed formats. Bookmark en-
66 tries in the PDF outline correspond to major document sections, facilitating
67 navigation in PDF readers.

68 **Supplementary Results**

69 This section presents additional validation results and performance bench-
70 marks for the SciTeX Writer framework that support the findings presented
71 in the main manuscript.

72 *Compilation Performance Benchmarks*

73 We measured compilation times across different system configurations to
74 assess the performance characteristics of the containerized compilation sys-
75 tem. On a reference system with 16 GB RAM and 8 CPU cores, compiling
76 the complete manuscript including all preprocessing steps required approx-
77 imately 12 seconds for the initial build and 4 seconds for subsequent incre-
78 mental builds when only content changed. The container startup overhead
79 added approximately 2 seconds to each compilation cycle. Compilation times
80 scaled linearly with document length, with the preprocessing pipeline con-
81 suming approximately 30% of total compilation time for documents with 10

82 or more figures. Parallel compilation of all three document types using `make`
83 `all` completed in approximately 18 seconds, demonstrating efficient resource
84 utilization through parallel processing.

85 *Cross-Platform Validation*

86 To verify true cross-platform reproducibility, we compiled identical source
87 files on six different system configurations spanning Ubuntu 20.04, macOS
88 13, Windows 11 with WSL2, CentOS 7, Arch Linux, and a high-performance
89 computing cluster running RHEL 8. Binary comparison of the resulting PDFs
90 using cryptographic hashing confirmed byte-for-byte identical outputs across
91 all platforms when using the same container image version. This valida-
92 tion extends to different processor architectures, with successful compilation
93 verified on both x86-64 and ARM64 systems. The only platform-specific
94 difference observed was in container startup time, which varied from 1.5 sec-
95 onds on native Linux to 3 seconds on macOS and WSL2 due to virtualization
96 overhead.

97 *Figure Format Conversion Validation*

98 The automatic figure processing system was validated with diverse in-
99 put formats including PNG, JPEG, SVG, PDF, TIFF, and EPS files. Fig-
100 ure ?? demonstrates the system’s handling of complex multi-panel figures
101 with mixed formats. Conversion quality was assessed by comparing pixel-
102 level differences between original and processed images. For lossless formats,
103 the conversion preserved perfect fidelity. For JPEG inputs, recompression
104 was avoided when possible to prevent quality degradation. SVG to PDF
105 conversion maintained vector properties, ensuring infinite scalability. Pro-
106 cessing times ranged from 0.1 seconds for simple PNG files to 2 seconds for
107 complex SVG graphics with extensive path data.

108 *Collaborative Workflow Testing*

109 We simulated collaborative editing scenarios by having multiple contrib-
110 utors simultaneously modify different manuscript sections in separate Git

111 branches. The modular file structure successfully prevented merge conflicts
112 in 94% of test cases involving concurrent edits. The remaining 6% of conflicts
113 occurred when contributors modified shared elements in the `00_shared/` di-
114 rectory, which is expected behavior. The shared metadata system correctly
115 propagated changes across all three document types in 100% of test cases.
116 Version archiving and difference generation performed correctly across branch
117 merges, maintaining complete history of document evolution.

118 *Scalability Analysis*

119 We tested the framework’s scalability by creating test documents rang-
120 ing from minimal manuscripts with 2 figures and 3 tables to comprehensive
121 documents with 50 figures, 30 tables, and over 100 pages of content. The
122 system handled all document sizes without modification to configuration or
123 structure. Memory consumption during compilation scaled linearly with doc-
124 ument size, requiring approximately 500 MB for minimal documents and 2.5
125 GB for the largest test cases. The modular architecture maintained orga-
126 nizational clarity even for complex documents, with navigation and editing
127 efficiency remaining constant across document sizes. Supplementary Table ??
128 provides detailed performance metrics across the tested range of document
129 complexities.

130 **Tables**

131 **Tables**

Table 1 – Table 0: Placeholder

To add tables to your manuscript:

- 1. Place CSV files in `caption_and_media/` with format `XX_description.csv`
- 2. Create matching caption files `XX_description.tex`
- 3. Reference in text using `Table~\ref{tab:XX_description}`

Example: `01_seizure_count.csv` with `01_seizure_count.tex`

Step	Instructions
1. Add CSV	Place file like <code>01_data.csv</code> in <code>caption_and_media/</code>
2. Add Caption	Create <code>01_data.tex</code> with table caption
3. Compile	Run <code>./compile -m</code> to process tables
4. Reference	Use <code>\ref{tab:01_data}</code> in manuscript

132 **Figures**

133 **Figures**

134 **References**