sciClaw: A Lightweight Paired-Scientist Agent for Reproducible Biomedical Research Workflows

Ernest Pedapati, MD

Cincinnati Children's Hospital Medical Center, Division of Child and Adolescent Psychiatry, Cincinnati, OH, USA

Corresponding author: Ernest Pedapati, MD, Cincinnati Children's Hospital Medical Center, 3333 Burnet Avenue, Cincinnati, OH 45229.

Structured Abstract

Objectives: Biomedical research workflows are fragmented across tools for literature search, data analysis, manuscript authoring, and experiment tracking, creating barriers to reproducibility. We developed sciClaw, a lightweight paired-scientist agent that integrates these capabilities into a single auditable loop, and evaluated it through a self-authoring case study and resource footprint analysis.

Materials and Methods: sciClaw extends PicoClaw, an open-source Go-based AI assistant, with a four-layer architecture: behavior templates, runtime context construction, a thirteen-skill research toolkit, and a governance layer with hardwired lifecycle hooks for loop-level traceability. We evaluated the system by using it to co-author this manuscript (literature search, drafting, tracked-change review, figure generation) and by comparing its resource footprint against OpenClaw, the dominant open-source agent platform.

Results: The paired scientist completed all research workflow tasks without requiring the researcher to write code or interact with external APIs directly. sciClaw adds thirteen research skills and seven governance hooks while maintaining a 17 MB binary and sub-50 ms startup time. A workflow capability comparison showed that sciClaw uniquely supports structured PubMed queries with RIS export, native Word documents with tracked changes, and cross-artifact provenance trailing, capabilities absent from general-purpose AI assistants.

Discussion: The paired-scientist model preserves human authority over scientific reasoning while delegating tool orchestration to the agent. Limitations include dependence on underlying LLM capabilities and the stochastic nature of model outputs.

Conclusion: sciClaw demonstrates that research-specific agent capabilities can be layered onto a minimal framework with negligible overhead, offering clinical and translational teams a practical path toward reproducible, auditable biomedical workflows.

Lay Summary

Biomedical researchers spend considerable time switching between different software tools: searching PubMed for papers, writing manuscripts in Word, managing references, and tracking their analyses. Each switch is an opportunity to lose track of what was done and why. We built sciClaw, a free, open-source AI research assistant that handles these tasks within a single conversation. The researcher gives directions in plain language (like talking to a colleague), and sciClaw searches the literature, drafts manuscript sections, creates properly formatted Word documents with tracked changes for collaborators, and exports references in formats that work with EndNote and Zotero. Everything the assistant does is automatically logged, so researchers can trace any result back to the exact search or analysis that produced it. sciClaw runs as a small program on the researcher's own computer, works with multiple AI providers (including Anthropic, OpenAI, and Google), and requires no programming

skills. We tested it by using it to help write this very paper, demonstrating the complete workflow from literature search through final manuscript formatting.

Background and Significance

Biomedical research demands coordination of multiple computational tools: literature retrieval from PubMed and preprint servers, statistical analysis, manuscript preparation, and version control. Every transition between tools introduces opportunities for error, lost provenance, and irreproducible shortcuts. The rise of large language model (LLM)-powered agents has created new possibilities for automating these transitions; however, existing systems are poorly suited to research workflows.

OpenClaw (a pseudonym used throughout this manuscript to refer to the dominant open-source personal AI agent platform), which amassed over 180,000 GitHub stars within weeks of its January 2026 launch, demonstrated demand for autonomous agents running on users' own hardware [1, 2]. Google's AI co-scientist [11] and OpenAI's Deep Research signaled that AI-assisted research was entering mainstream discourse. However, these systems either target general consumer automation or operate as cloud-hosted services with limited access to paywalled literature and no local tool integration. OpenClaw's TypeScript runtime demands over 1 GB of RAM with no built-in support for literature search or manuscript co-authoring. Pipeline tools like Nextflow [14] and Snakemake [15] automate computational workflows but lack interactive, reasoning-driven capabilities.

We propose the concept of the **paired scientist**, analogous to pair programming in software engineering [12]. The human researcher serves as navigator (framing questions, evaluating evidence, making intellectual judgments) while the AI agent serves as driver (executing searches, running analyses, drafting manuscripts, maintaining the evidence trail). This preserves human authority over scientific reasoning while delegating mechanical tool orchestration to the agent. To our knowledge, the term “paired scientist” hasn’t appeared in prior literature; we introduce it to distinguish this collaborative model from Google’s “co-scientist” paradigm [11], which emphasizes autonomous hypothesis generation.

Recent work on LLM agents for scientific workflows includes the EXHYTE framework formalizing discovery cycles [1], multi-agent systems for earth science [2], retrieval-augmented strategies for materials design [3], and Biomni for biomedical task execution [4]. Domain-specific applications have extended these approaches to drug discovery pipelines [5], bioinformatics benchmarking [6], radiology workflow governance [8], and broader healthcare agent architectures [10]. In reproducibility, dtoolAI implements automatic provenance capture aligned with FAIR principles [7], Vitlov et al. articulate how executable workflows and provenance-preserving computation are redefining responsible data sharing under FAIR guidelines [9], and Scherbakov et al. found that 73% of LLM-assisted literature reviews use GPT-based architectures but focus on individual stages rather than end-to-end workflows [13]. sciClaw differs by operating as a single lightweight binary that integrates research tools directly into the agent loop while preserving upstream compatibility.

A central design tension is that researchers are deeply technical users (expert in biostatistics, clinical trial design, and domain-specific methodologies such as neuroimaging or genomics) but aren’t necessarily programmers. They can reason about MeSH hierarchies, critique study designs, and interpret complex biological data, yet may not be comfortable writing Python scripts or debugging API integrations. This distinction pervades sciClaw’s design: workspace behavior is defined in Markdown files any researcher can edit, skills are authored as natural-language specifications rather than code, and the agent is accessed through familiar messaging applications (Discord, Telegram, Slack) rather than a command-line terminal.

A critical gap in existing AI assistants concerns the workflows biomedical researchers actually use daily. Scientists routinely query PubMed with MeSH terms, fetch structured metadata by PMID, export citations in RIS format for reference managers, and trace citation graphs to map a field. Manuscript review in biomedicine revolves around Microsoft Word documents with tracked changes and reviewer comments, not Markdown or web-based editors. No current general-purpose AI assistant supports structured PubMed queries, citation graph traversal, RIS export, or native Word document creation with tracked changes. sciClaw addresses this through dedicated CLI tools as primitive building blocks. pubmed-cli provides native PubMed search, article fetch, citation graph traversal, MeSH lookup, and RIS export. docx-review enables Word document creation and editing with tracked changes, anchored comments, and semantic diffing. These tools are surfaced to the agent through the Anthropic Agent Skills standard, an open specification released in December 2025, where each skill is a SKILL.md file containing YAML frontmatter and Markdown instructions that domain experts can author without writing code.

Security is a systemic challenge for autonomous agents. OpenClaw's rapid adoption has been accompanied by security incidents including CVE-2026-25253 (one-click RCE, CVSS 8.8), over 21,000 publicly exposed instances leaking credentials, and 341 malicious skills on its community registry. sciClaw adopts a defense-in-depth approach: workspace sandboxing, messaging channel allow-lists, SHA-256 checksums for skill integrity verification, and the governance layer's unconditional audit logging. Credential management is deliberately separated from workspace templates to prevent accidental exposure.

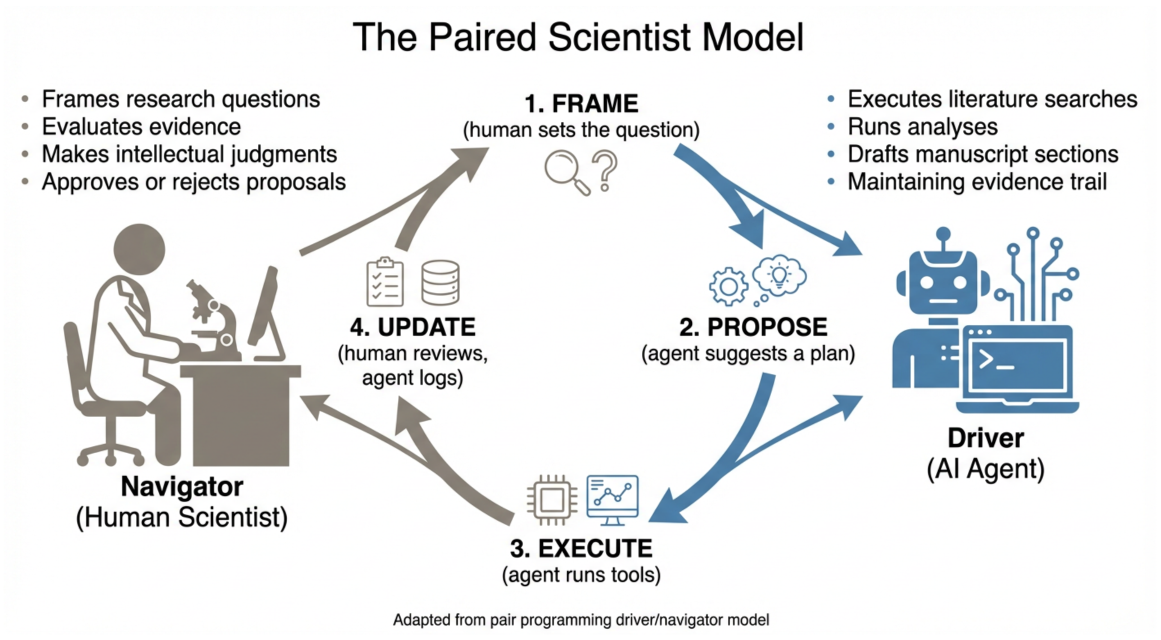
The Paired Scientist Model
Frames research questions
Evaluates evidence
Makes intellectual judgments
Approves or rejects proposals
1. FRAME
(human sets the question)
Executes literature searches
Runs analyses
Drafts manuscript sections
Maintaining evidence trail
4. UPDATE
(human reviews, agent logs)
2. PROPOSE
(agent suggests a plan)
Driver
(AI Agent)
Navigator
(Human Scientist)
3. EXECUTE
(agent runs tools)
Adapted from pair programming driver/navigator model

Figure 1. The paired scientist model. The human researcher serves as navigator while the AI agent serves as driver. The four-stage loop (Frame, Propose, Execute, Update) maintains continuous collaboration with human authority over scientific reasoning.

Objectives

This work addresses three objectives: (1) design and implement a paired-scientist agent architecture that integrates literature search, manuscript authoring, document review, and experiment provenance into a single reproducible workflow; (2) evaluate the system through a self-authoring case study demonstrating end-to-end research task completion; and (3) quantify the resource overhead of adding research-specific capabilities to a minimal agent framework.

Materials and Methods

System Architecture

sciClaw extends PicoClaw, an ultra-lightweight Go-based AI assistant developed by Sipeed, with a four-layer architecture (Figure 2):

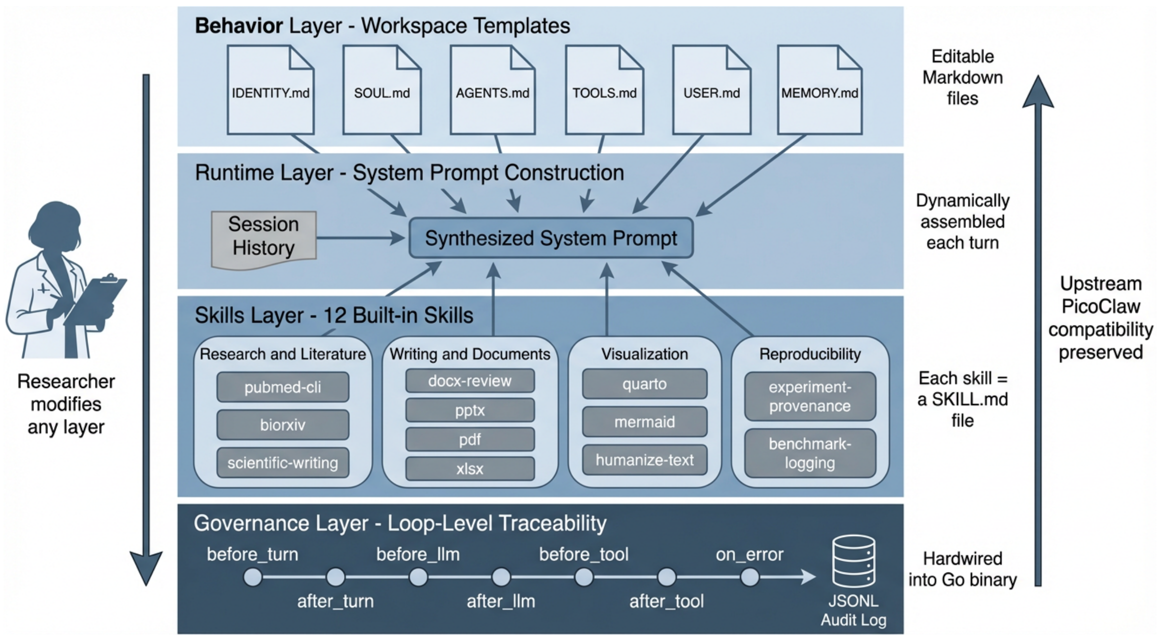

Figure 2. sciClaw four-layer architecture. The Behavior Layer defines agent identity through editable Markdown files. The Runtime Layer dynamically constructs system prompts. The Skills Layer provides thirteen research capabilities. The Governance Layer hardwires seven lifecycle hooks producing append-only JSONL audit logs.

Behavior Layer. Six Markdown template files (IDENTITY.md, SOUL.md, AGENTS.md, TOOLS.md, USER.md, MEMORY.md) define the agent's identity, values, operating protocol, and user preferences, editable by any researcher without code changes.

Runtime Layer. A context builder dynamically synthesizes the system prompt from workspace templates, skill documentation, session history, and persistent state.

Skills Layer. Thirteen biomedical research skills organized into five categories: *Research and Literature* (pubmed-cli, biorxiv-database, scientific-writing); *Writing and Documents* (docx-review, pptx, pdf, xlsx); *Visualization and Authoring* (quarto-authoring, beautiful-mermaid, humanize-text); *Clinical Data* (phi-cleaner); *Reproducibility* (experiment-provenance, benchmark-logging).

Governance Layer. Seven lifecycle hook events (before_turn, after_turn, before_llm, after_llm, before_tool, after_tool, on_error) are hardwired into the Go binary. Each fires with an immutable context snapshot capturing turn ID, model, tool name, arguments, results, and timing. Append-only JSONL audit entries are written unconditionally, with automatic redaction of sensitive fields (Figure 3).

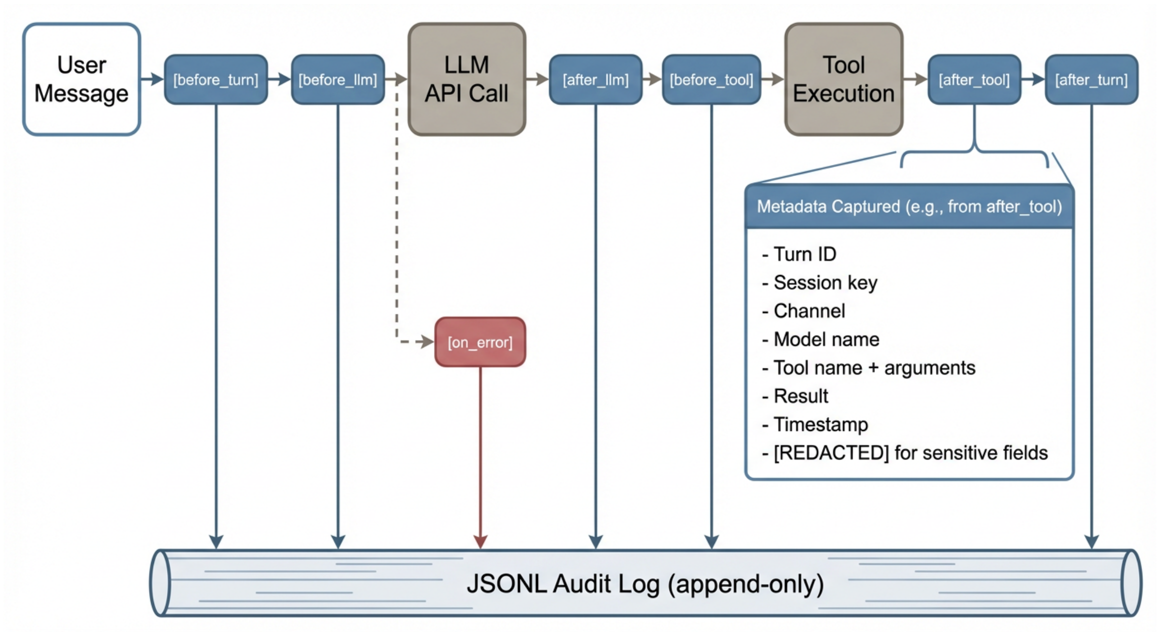

Figure 3. Governance and audit trail flow. Seven lifecycle hooks fire at each stage of the agent turn, writing immutable snapshots to the append-only JSONL audit log. Sensitive fields are automatically redacted.

A provider-agnostic LLMProvider interface abstracts model interactions, with implementations for Anthropic, OpenAI, Google Gemini, Groq, DeepSeek, and others. Model resolution is automatic from name prefix (e.g., claude-* routes to Anthropic). Reasoning effort is configurable per task (six levels from *none* through *xhigh* for GPT-5.2). The self-authoring case study described below used OpenAI GPT-5.2 as the primary LLM, accessed through sciClaw's provider abstraction.

A *sciclaw service* command manages the messaging gateway as a background service using platform-native init systems: launchd on macOS and systemd --user on Linux. Operations include install, uninstall, start, stop, restart, status, and log viewing. The service auto-generates platform-appropriate configuration files (a LaunchAgent plist on macOS, a systemd user unit on Linux), with graceful degradation and guidance for unsupported environments such as WSL without systemd.

Companion CLI Tools

Three of sciClaw's skills depend on custom-developed companion CLI tools distributed as standalone binaries via Homebrew:

pubmed-cli (https://github.com/drpedapati/pubmed-cli): A Go binary providing structured PubMed search with MeSH term expansion, article fetch by PMID, citation graph traversal (cited-by and references), related-article discovery, MeSH hierarchy lookup, and RIS/CSV export. It implements direct NCBI E-utilities API integration with automatic rate limiting, structured JSON output, and human-readable formatting.

docx-review (https://github.com/drpedapati/docx-review): A C#/.NET 8 binary that adds tracked changes and anchored comments to existing Microsoft Word documents via JSON manifests. It supports semantic diffing between document versions, document content extraction (read mode), and a git textconv driver for version-controlled .docx files. This tool bridges the gap between AI-generated edits and the tracked-change review workflow standard in academic publishing.

phi-cleaner (https://github.com/drpedapati/phi-cleaner): A Python tool using BERT-based named entity recognition models to de-identify clinical text by detecting and redacting protected health information (PHI) including patient names, dates, provider names, and institutional identifiers. It enables researchers to share clinical notes and free-text data while maintaining HIPAA compliance, a prerequisite for reproducible clinical research.

A fourth companion tool, **IRL** (https://github.com/drpedapati/irl-template), is a Go binary providing Idempotent Research Loop project lifecycle management with auto-naming from purpose strings (YYMMDD-slug format), template-based scaffolding, and git-based version control. All four tools are installed automatically as Homebrew dependencies and verified by the *sciclaw doctor* command.

Original Contributions

Table 3 provides a comprehensive inventory of sciClaw's modifications and additions relative to the PicoClaw base system. All components listed represent original work developed for this project.

Category	Components	Original Work
Governance system	5 Go packages, 7 hook events	Lifecycle hooks hardwired into binary, JSONL audit logger, plain-language and YAML policy engine, provenance handler, automatic secrets redaction
CLI commands	8 new commands	doctor (deployment verification), service (background lifecycle), channels (setup wizards), models (switching + reasoning effort), auth (OAuth device-code and token-paste), onboard (interactive first-run), backup, migrate
Companion CLI tools	4 custom-developed tools	pubmed-cli (Go; PubMed search, citation graphs, RIS export), docx-review (C#/.NET 8; Word tracked changes, comments, semantic diff), phi-cleaner (Python; BERT-based clinical text de-identification), IRL (Go; project lifecycle, template scaffolding)
Research skills	13 SKILL.md files	pubmed-cli, biorxiv-database, scientific-writing, docx-review, pptx, pdf, xlsx, quarto-authoring, beautiful-mermaid, humanize-text, phi-cleaner, experiment-provenance, benchmark-logging
Workspace templates	7 Markdown files	IDENTITY.md, SOUL.md, AGENTS.md, HOOKS.md, TOOLS.md, USER.md, MEMORY.md
Provider system	6 LLM backends	Anthropic (SDK + OAuth), OpenAI (streaming + device-code), Google Gemini, Groq (+ Whisper voice), DeepSeek, OpenRouter; per-task reasoning effort
Service lifecycle	3 platform backends	launchd (macOS), systemd --user (Linux), graceful WSL fallback
New Go packages	15 packages	hooks, hookpolicy, service, irl, heartbeat, cron, voice, bus, session, state, migrate, skills, models, auth, logger
Channel enhancements	8 messaging platforms	Telegram (+ voice transcription), Discord (+ typing indicators), Slack, QQ, DingTalk, Feishu, LINE, WhatsApp; interactive setup wizards
Security hardening	6 measures	Config permissions (0600), skill catalog pinned to immutable commit refs, SHA-256 checksums, channel allow-lists, automatic secrets redaction, heartbeat disabled by default
Infrastructure	4 components	Homebrew formula with dependency chain, Docker Compose profiles, CI/CD release automation, documentation site

Table 3. Comprehensive inventory of sciClaw modifications and additions relative to PicoClaw base. All items represent original work developed for this project.

Self-Authoring Case Study

To demonstrate the paired-scientist workflow in a realistic setting, we used sciClaw to co-author this manuscript. The researcher served as navigator throughout, framing the argument, selecting content, evaluating drafts, and making editorial decisions. The agent served as driver, executing six task categories under human direction:

1. **Literature search.** The pubmed-cli skill searched PubMed across four query domains (agentic AI for scientific discovery, domain-specific agents, reproducibility frameworks, and pair programming). The agent retrieved candidate papers, fetched abstracts, and presented ranked results. The researcher selected 12 references (later expanded to 13) for inclusion.
2. **Manuscript drafting.** Starting from a bullet-point outline in a Quarto (.qmd) file, the agent expanded each section into prose, producing a complete Markdown draft. The researcher reviewed each section and directed revisions across multiple conversation turns.
3. **Document review.** The docx-review skill generated a formatted Word document with tracked changes, enabling the researcher to accept, reject, or modify edits using standard Word review tools. Two rounds of tracked-change edits covered narrative framing, security analysis, and governance details.
4. **Language polishing.** The humanize-text skill performed a final editorial pass, removing AI writing artifacts while preserving academic register. Word counts were compared before and after to verify no content loss.
5. **Figure generation.** The agent authored detailed figure briefs for illustrations, then generated draft figures through an image generation API. The researcher reviewed each for accuracy.

6. **Iterative refinement.** The researcher directed additional content and revised the evaluation framing through conversational instructions. Each change was applied to both Markdown source and Word document, maintaining consistency.

All agent actions were logged through the governance layer's JSONL audit trail.

Representative Research Workflows

We identified three workflows central to biomedical research that sciClaw supports natively but remain fragmented across existing tools:

Workflow A (Literature Review): PubMed search with MeSH terms and Boolean operators, structured metadata retrieval, citation graph traversal, and RIS export for EndNote/Zotero, all within a single conversation.

Workflow B (Manuscript with Tracked Changes): Markdown drafting, formatted .docx generation with academic styling, and tracked-change revisions reviewable in Microsoft Word. No current AI assistant bridges AI-generated content and the tracked-change review workflow standard in academic publishing.

Workflow C (Multi-Format Output): Single-session generation of Word manuscript, PowerPoint slides, Excel data summary, and RIS citations, with cross-artifact provenance logging.

Resource Footprint Analysis

We measured binary size, cold-start time, peak memory, dependency count, and skill count for sciClaw, PicoClaw (upstream base), and OpenClaw on an Apple M3 Max workstation with 64 GB unified memory running macOS 15.6.

Results

Case Study Outcomes

The paired scientist completed all six task categories without requiring the researcher to write code, run shell commands, or interact with external APIs directly. The researcher's role was limited to intellectual direction: framing questions, evaluating outputs, and approving or rejecting proposed changes.

Provenance continuity. The governance layer captured every tool invocation across the manuscript's development. Any sentence in the final document can be traced through the JSONL log to the specific conversation turn, LLM call, and tool execution that produced it, including the literature search queries that identified each reference and the docx-review commands that applied tracked changes.

Iterative convergence. The manuscript required multiple revision cycles, each initiated through natural-language instructions ("add discussion about secret management," "rewrite the methods section"). The agent applied changes to both Markdown source and Word document in a single pass, maintaining consistency between representations. This conversational refinement pattern reflects the navigator/driver dynamic.

Self-referential bootstrapping. The agent used its pubmed-cli skill to find references for the Background and Significance section, its docx-review skill to format the document it describes, and its humanize-text skill to polish the prose explaining how skills work. If the system can produce a publishable manuscript about itself, it should be capable of producing manuscripts about other research topics using the same workflow.

Workflow Capability Comparison

Table 1 compares sciClaw against general-purpose AI assistants (ChatGPT, Claude, Gemini) and cloud research tools (Google AI co-scientist, Elicit, Semantic Scholar) across representative research workflow capabilities.

Capability	General-purpose AI	Cloud research tools	sciClaw
PubMed structured query (MeSH)	No	Limited	Yes
Citation graph traversal	No	No	Yes
RIS export for reference managers	No	No	Yes
Native .docx with tracked changes	No	No	Yes
Multi-format output (docx+pptx+xlsx)	No	No	Yes
Local execution	No	No	Yes
Clinical text de-identification (PHI)	No	No	Yes
Cross-artifact provenance trail	No	No	Yes

Table 1. Workflow capability comparison across existing AI tools and sciClaw.

The most pronounced gap is in manuscript authoring with tracked changes. No current AI product produces a Word document with tracked changes reviewable in Microsoft Word, despite this being a routine workflow for academic researchers.

Resource Footprint

Table 2 compares resource characteristics across agent platforms.

Metric	OpenClaw	PicoClaw	sciClaw
Language	TypeScript	Go	Go
Install size	~500 MB	~15 MB	17 MB
Cold-start time	>60 s	<50 ms	<50 ms
Peak memory	>1 GB	<10 MB	<15 MB
Dependencies	~1,200 (npm)	~30 (Go)	38 (Go)
Research skills	0	0	13
Governance hooks	0	0	7 (hardwired)

Table 2. Resource footprint comparison. sciClaw adds research capabilities with negligible overhead.

Research-specific capabilities increased binary size by only 2 MB over baseline PicoClaw, reflecting additional Go code for hook handling, provider routing, and project lifecycle management. Skill definitions (SKILL.md files) are loaded on demand and don’t contribute to binary size. Installation requires a single Homebrew command (*brew install* *drpedapati/tap/sciclaw*) with no additional runtime dependencies, Docker containers, or cloud infrastructure.

Discussion

sciClaw demonstrates that a paired-scientist architecture can be implemented as a lightweight behavioral extension of an existing open-source agent framework. The four-layer design allows researchers to customize the agent through Markdown files rather than code, accommodating life scientists who are expert in their domains but may not be programmers.

The self-authoring case study provides practical evidence that the paired-scientist workflow can produce a complete manuscript, from literature search through formatted Word document, within a single auditable session. The governance layer's unconditional logging creates provenance records that support reproducibility at the tool-execution level. The bootstrapping property, where sciClaw uses its own skills to build the manuscript that describes those skills, demonstrates that the system's capabilities generalize beyond any single research domain.

Several limitations merit discussion. The system's effectiveness depends on the underlying LLM's capabilities; model-specific failure modes (hallucination, reasoning errors, tool misuse) propagate through the agent loop. The provider abstraction mitigates vendor lock-in but doesn't solve capability gaps. The reproducibility guarantees apply to the tool execution trail but not to stochastic LLM outputs; identical prompts may produce different responses across runs. A significant limitation of the current evaluation is that the system developer served as the sole user and evaluator of the self-authoring case study. This introduces potential bias: the developer's familiarity with the system's internal architecture, command syntax, and failure modes likely produced more effective navigator behavior than would be expected from a naive user. Workflow friction, error recovery patterns, and learning curve effects that independent researchers would encounter are not captured. Planned next steps include structured user studies with clinical and translational researchers across multiple biomedical institutions, using standardized task suites and think-aloud protocols to assess usability, task completion rates, and perceived utility independent of developer expertise.

Security considerations include the defense-in-depth approach inherited from PicoClaw's minimal surface area, workspace sandboxing, messaging channel allow-lists, and SHA-256 checksums for skill integrity verification. Secret management is addressed by separating credential storage from workspace templates: API keys are read from environment variables or the system keychain, never written to workspace files.

A practical concern for agent-based workflows is secret management. Agents need credentials: LLM provider keys, PubMed API tokens, messaging bot tokens. Scientists accustomed to pasting API keys into configuration files adopt practices that don't scale safely. sciClaw reads credentials from environment variables or the system keychain, never writing them to workspace files that might be committed to version control. A *sciclaw doctor* command validates credential security and checks for required companion tools.

The architecture is designed to accommodate a future in which local LLM capabilities approach cloud model performance. The same LLMProvider interface that routes to Anthropic or OpenAI can route to a local model via Ollama, enabling fully private operation without changing skills or workflows. The paired scientist's competence also grows incrementally: the *sciclaw skills install* command fetches new skill definitions, validates their SKILL.md structure, records provenance metadata (source URL, SHA-256 hash, timestamp), and makes them available in the next conversation turn.

Conclusion

We have presented sciClaw, a paired-scientist agent extending PicoClaw with thirteen research skills, seven hardwired governance hooks, and a provider-agnostic model abstraction, all within a 17 MB binary requiring no coding expertise. The self-authoring case study and workflow analysis demonstrate practical utility for literature search, manuscript authoring with tracked changes, and reproducible research documentation. For clinical and translational teams, sciClaw offers auditable, reproducible workflows installable via a single command. Future work will focus on user studies across diverse biomedical research domains, expanding the skill library to additional life science databases and analysis tools, and validating the paired-scientist model at scale in clinical and translational settings.

Data Availability

sciClaw is open source under the MIT license. Source code is available at

Repository: https://github.com/drpedapati/sciclaw

Upstream base: https://github.com/sipeed/picoclaw

The governance layer audit logs generated during this manuscript's preparation are available in the project repository.

Funding

None declared.

Competing Interests

The author declares no competing interests.

Author Contributions

Ernest Pedapati: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft, Writing - Review & Editing, Visualization.

Acknowledgments

This manuscript was co-authored using sciClaw's paired-scientist workflow, with the author serving as navigator and the sciClaw agent serving as driver. The pubmed-cli, docx-review, and humanize-text skills were used during manuscript preparation.

References

[1] Hasib MM, Jo S, Sinha H, et al. A Process-Centric Survey of AI for Scientific Discovery Through the EXHYTE Framework. *Res Sq.* 2025. doi:10.21203/rs.3.rs-8370059/v1. PMID: 41510245.

[2] Pantiukhin D, Shapkin B, Kuznetsov I, Jost AA, Koldunov N. Accelerating earth science discovery via multi-agent LLM systems. *Front Artif Intell.* 2025;8:1674927. doi:10.3389/frai.2025.1674927. PMID: 41312024.

[3] Buehler MJ. Generative Retrieval-Augmented Ontologic Graph and Multiagent Strategies for Interpretive Large Language Model-Based Materials Design. *ACS Eng Au.* 2024;4(2):241-277. doi:10.1021/acsengineeringau.3c00058. PMID: 38646516.

[4] Huang K, Zhang S, Wang H, et al. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv.* 2025. doi:10.1101/2025.05.30.656746. PMID: 40501924.

[5] Vichentijevikj I, Mishev K, Simjanoska Misheva M. Prompt-to-Pill: Multi-Agent Drug Discovery and Clinical Simulation Pipeline. *Bioinform Adv.* 2026;6(1):vbaf323. doi:10.1093/bioadv/vbaf323. PMID: 41542364.

[6] Rajesh V, Siwo GH. Out-of-the-box bioinformatics capabilities of large language models (LLMs). *bioRxiv.* 2025. doi:10.1101/2025.08.22.671610. PMID: 40909484.

[7] Hartley M, Olsson TSG. dtoolAI: Reproducibility for Deep Learning. *Patterns (N Y).* 2020;1(5):100073. doi:10.1016/j.patter.2020.100073. PMID: 33205122.

[8] Tzanis E, Adams LC, Akinci D'Antonoli T, et al. Agentic systems in radiology: Principles, opportunities, privacy risks, regulation, and sustainability concerns. *Diagn Interv Imaging.* 2026;107(1):7-16. doi:10.1016/j.diii.2025.10.002. PMID: 41432042.

[9] Vitlov N, Vukovic M, Bralic N, Marusic A. Waste Not, Want Not: How to Make Your Data Futureproof Through Good Data Sharing Practices. *Curr Protoc.* 2025;5(12):e70283. doi:10.1002/cpz1.70283. PMID: 41427491.

[10] Srinivasu PN, Aruna Kumari GL, Ahmed S, Alhumam A. Exploring Agentic AI in Healthcare: A Study on Its Working Mechanism. *Front Med (Lausanne).* 2025;12:1753443. doi:10.3389/fmed.2025.1753443. PMID: 41685258.

[11] Moons P, Dou B, Desmedt CP. Google's AI co-scientist and OpenAI's deep research: new partners in health research? *Eur J Cardiovasc Nurs.* 2025;24(5):800-807. doi:10.1093/eurjcn/zvaf066. PMID: 40678974.

[12] Valovy M, Buchalcevova A. Personality-based pair programming: toward intrinsic motivation alignment in very small entities. *PeerJ Comput Sci.* 2025;11:e2774. doi:10.7717/peerj-cs.2774. PMID: 40567791.

[13] Scherbakov D, Hubig N, Jansari V, Bakumenko A, Lenert LA. The emergence of large language models as tools in literature reviews: a large language model-assisted systematic review. *J Am Med Inform Assoc.* 2025;32(6):1071-1086. doi:10.1093/jamia/ocaf063. PMID: 40332983.

[14] Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. *Nat Biotechnol.* 2017;35(4):316-319. doi:10.1038/nbt.3820. PMID: 28398311.

[15] Mölder F, Jablonski KP, Letcher B, et al. Sustainable data analysis with Snakemake. *F1000Res.* 2021;10:33. doi:10.12688/f1000research.29032.3. PMID: 34035898.

Supplemental Data: Reproducibility Audit of the Self-Authoring Case Study

Manuscript: sciClaw: An Autonomous Paired Scientist for Reproducible Scientific Workflows
Author: Ernest Pedapati, Cincinnati Children's Hospital Medical Center **Date:** 2026-02-19

S1. Overview

The manuscript describes sciClaw's paired-scientist architecture and presents a self-authoring case study (Section 4.1) in which the system was used to co-author its own manuscript. This supplemental document provides the quantitative data from that process, organized as a reproducibility audit.

All data were extracted from artifacts produced during the manuscript authoring sessions: Claude Code session transcripts (JSONL), tracked-change edit records (JSON), a PubMed reference verification report, git commit history, and sciClaw governance hook event logs. The analysis scripts that produced these results are included in this directory and can be re-executed to reproduce all tables and figures.

S2. Interaction Metrics

Table S1. Session-Level Metrics

The manuscript was authored across two interactive sessions using Claude Code as the human-side interface to the paired scientist.

Metric	Session 1 (Feb 14-15)	Session 2 (Feb 15-16)	Total
Total session records	3,265	3,236	6,501
Human messages	664	526	1,190
Agent responses	1,043	829	1,872
Tool invocations	616	492	1,108
Substantial responses (>100 chars)	164	119	283
Session duration (hours)	85.8*	15.6	–

*Session 1 duration reflects wall-clock time including overnight idle periods; active working time was substantially less.

Table S2. Tool Usage Breakdown

Tool	Count	%	Role in Workflow
Bash	427	38.5%	Shell commands: git operations, build scripts, file management
Edit	265	23.9%	Targeted modifications to manuscript source and build scripts

Tool	Count	%	Role in Workflow
Read	187	16.9%	Reading manuscript sections, source code, reference data
Grep	75	6.8%	Searching codebase for implementation details and patterns
Write	37	3.3%	Creating new files: build scripts, figure briefs, edit records
TaskUpdate	28	2.5%	Tracking multi-step workflow progress
Task	23	2.1%	Spawning sub-agents for parallel research
WebSearch	19	1.7%	Verifying facts, searching for related work, checking CVEs
TaskCreate	14	1.3%	Planning workflow phases
WebFetch	12	1.1%	Fetching web page content for fact verification
Glob	9	0.8%	File discovery across project directories
ExitPlanMode	8	0.7%	Transitioning from planning to execution phases
EnterPlanMode	2	0.2%	Initiating structured planning
Skill	1	0.1%	Invoking specialized skill (docx creation)
AskUserQuestion	1	0.1%	Clarifying human researcher intent
Total	**1,108**		

The tool distribution reveals a workflow dominated by iterative file manipulation (Edit: 23.9%) and shell execution (Bash: 38.5%), with substantial reading (Read: 16.9%) reflecting the agent's need to understand existing content before modifying it. Web-based research tools (WebSearch + WebFetch) account for only 2.8% of invocations, suggesting that the bulk of literature search was completed in earlier sessions before manuscript drafting began. The low Skill invocation count (1) reflects that the docx-review skill was invoked through Bash commands rather than the dedicated Skill tool.

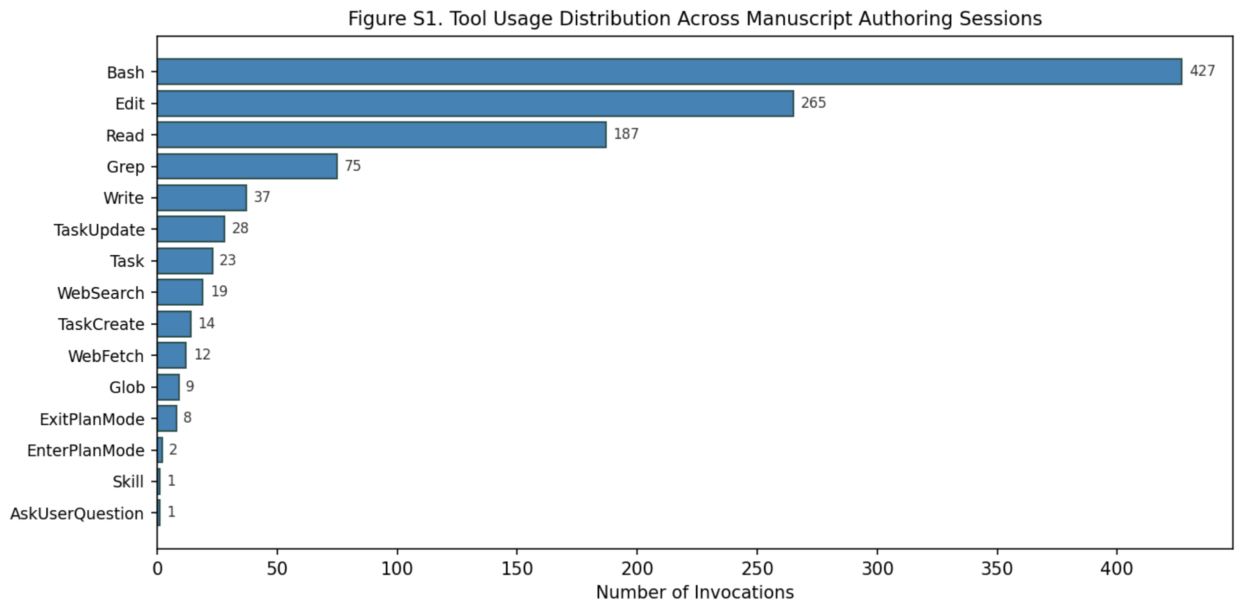

Figure S1. Tool usage distribution across manuscript authoring sessions. Bash and Edit dominate, reflecting an iterative write-review-revise workflow.

S3. Revision Process

Table S3. Tracked Changes Summary

Two rounds of structured tracked changes were applied to the manuscript, recorded as JSON edit specifications that the docx-review skill translated into Word tracked changes.

Metric	Round 1	Round 2	Total
Changes applied	3	4	7
Reviewer comments	8	2	10
Words removed	150	127	277
Words added	1,667	496	2,163
Net words	+1,517	+369	+1,886

Round 1 was expansive: it replaced the original brief introduction with a comprehensive narrative covering OpenClaw, the paired-scientist concept, security analysis, and messaging architecture. The largest single change replaced 52 words with 1,157 words, transforming a two-sentence gap statement into a multi-paragraph introduction covering PicoClaw, the scientist-as-technical-user design insight, CLI tool philosophy, Agent Skills standard, security incidents, and messaging-first interaction.

Round 2 was focused: it refined the governance layer description (replacing a general summary with specific hook implementation details), expanded the IRL integration section, and added context about reasoning effort configuration. Changes in Round 2 averaged +94 net words versus +506 in Round 1.

Table S4. Comment Classification

Round	Category	Count	Description
Round 1	Factual note	4	External facts requiring verification (OpenClaw history, skill standard timeline, bootstrapping claim, local model support)
Round 1	Novel term flag	1	Flagging "paired scientist" as a novel coinage not found in prior PubMed literature
Round 1	Security note	1	CVE-2026-25253 details, CVSS score, exposed instance count
Round 1	Source citation	1	New reference insertion for Google co-scientist and pair programming papers
Round 1	Design insight	1	Annotation explaining the scientist-as-technical-user design principle

Round	Category	Count	Description
Round 2	Factual note	2	Source code file paths for hook implementation and IRL client

Comments served two distinct functions: (a) providing the human researcher with context for evaluating proposed changes (factual notes, security notes), and (b) flagging claims that required explicit human judgment (novel term flag, design insight). This reflects the navigator/driver dynamic: the agent annotates its proposed changes so the human can make informed accept/reject decisions.

S4. Reference Verification

Table S5. Reference Audit Results

All 15 references in the manuscript were verified against the live NCBI PubMed database using `pubmed fetch <PMID> --json` (pubmed-cli v0.5.4). Each field was compared: authors, title, journal, year, volume, issue, pages, and DOI.

Ref	Status	Severity	Error Details
[1] Hasib et al.	PASS	–	–
[2] Pantiukhin et al.	PASS	–	–
[3] Buehler	PASS	–	–
[4] Huang et al.	PASS	–	–
[5] Vichenti-jevikj et al.	PASS	–	–
[6] Rajesh & Siwo	PASS	–	–
[7] Hartley & Olsson	PASS	–	–
[8] Tzanis et al.	PASS	–	–
[9] Vitlov et al.	PASS	–	Diacritics stripped (acceptable)
[10] Srinivasu et al.	PASS	–	–
[11] Moons et al.	FAIL	Minor	Wrong author initials: “Dou R” should be “Dou B”; “Desmedt M” should be “Desmedt CP”
[12] Valovy & Buchal-cevova	FAIL	Minor	Wrong author initial: “Valovy D” should be “Valovy M”

Ref	Status	Severity	Error Details
[13] Scherbakov et al.	FAIL	Critical	Hallucinated author: "Hubner J" should be "Hubig N"; "Gartlehner G" not in author list; wrong page range: 1083 should be 1086
[14] Di Tommaso et al.	PASS	–	–
[15] Molder et al.	FAIL	Minor	Wrong DOI version: ".2" should be ".3"

Summary: 11/15 references (73.3%) passed verification. 4/15 (26.7%) contained errors. One reference ([13]) had a critical error: the LLM fabricated an author name ("Hubner J" instead of "Hubig N") and inserted an author ("Gartlehner G") who does not appear on the paper at all.

Error taxonomy: - Wrong author initials: 3 instances across 2 references - Hallucinated author name: 1 instance (surname fabricated) - Hallucinated author (not on paper): 1 instance - Wrong page range: 1 instance - Wrong DOI version: 1 instance

Context. The 26.7% citation error rate is consistent with published findings on LLM citation reliability. Scherbakov et al. [13] (ironically, the reference with the most errors in our audit) systematically reviewed LLM performance in literature review tasks and found that citation accuracy remains a significant limitation, with models frequently producing plausible but incorrect bibliographic metadata. This finding underscores the necessity of automated verification tools like pubmed-cli for any LLM-assisted citation workflow: the paired scientist can search, retrieve, and format citations efficiently, but a human or automated verification step is essential before publication.

All 4 errors were corrected in the final manuscript based on this audit.

S5. Development Timeline

Table S6. Git Commit History

The manuscript directory was developed across 16 git commits spanning 20.1 hours (Feb 12-13, 2026).

Phase	Time Range	Commits	Category	Description
1	Feb 12, 07:58	1	Scaffolding	Initial plan execution, manuscript outline
2	Feb 12, 08:57-13:00	9	Run-loop documentation	Iterative planning and documentation cycles
3	Feb 13, 03:04-03:44	5	Implementation	Skills revision, branding, binary rebrand

Phase	Time Range	Commits	Category	Description
4	Feb 13, 04:05	1	Release automation	Homebrew and container packaging

Commit category breakdown: - Run-loop documentation: 10 (62.5%) – iterative documentation and planning - Implementation: 3 (18.8%) – code and configuration changes - Skills integration: 1 (6.3%) - Documentation/UI: 1 (6.3%) - Scaffolding: 1 (6.3%)

The commit pattern shows two concentrated burst periods (Feb 12 09:00 with 6 commits, Feb 13 03:00 with 5 commits) separated by an overnight gap, suggesting two focused working sessions rather than continuous development.

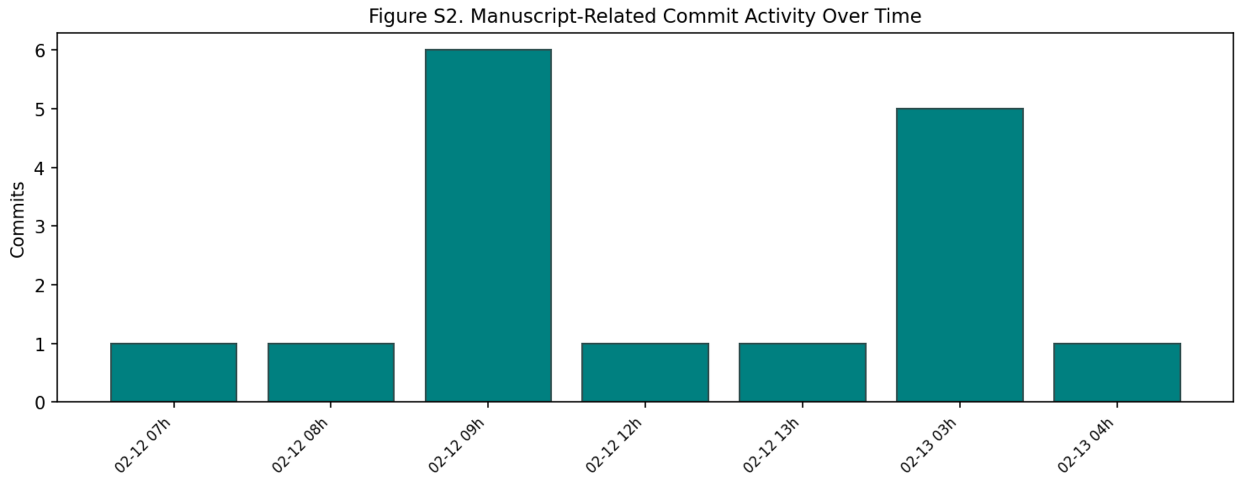

Figure S2. Manuscript-related commit activity over time, showing two concentrated burst periods.

S6. Governance Architecture Demonstration

Table S7. Governance Event Structure

sciClaw's governance layer fires seven lifecycle hook events per agent turn: `before_turn`, `after_turn`, `before_llm`, `after_llm`, `before_tool`, `after_tool`, and `on_error`. Two built-in handlers (provenance and policy) execute on each event, producing immutable JSONL audit entries.

The following trace shows the complete event sequence for a single agent turn captured during a sciClaw demonstration session:

#	Event	Handler	Status	Message	Timestamp
1	before_turn	provenance	ok	provenance captured	2026-02-17T01:40:27.402
2	before_turn	policy	ok	Capture project context needed for reproducibility	2026-02-17T01:40:27.402

#	Event	Handler	Status	Message	Timestamp
3	before_llm	provenance	ok	provenance captured	2026-02-17T01:40:27.404
4	before_llm	policy	ok	Note important constraints or safety boundaries	2026-02-17T01:40:27.404
5	after_llm	provenance	ok	provenance captured	2026-02-17T01:40:28.207
6	after_llm	policy	ok	Capture a concise response summary and confidence caveats	2026-02-17T01:40:28.207
7	after_turn	provenance	ok	provenance captured	2026-02-17T01:40:28.229
8	after_turn	policy	ok	Summarize completed actions and any unresolved risks	2026-02-17T01:40:28.229

Metadata fields captured per event: event, event_enabled, event_metadata (iteration count, messages count, tools count, tool call count), instructions, policy_enabled, session_key, turn_id, verbosity.

Provenance handler normalizes event metadata and writes raw context snapshots. **Policy handler** evaluates workspace-defined rules and injects context-specific instructions (e.g., "Capture project context needed for reproducibility" at turn start, "Summarize completed actions and any unresolved risks" at turn end).

Note: This trace captures 2 demonstration turns (16 events total). The demonstration turns did not invoke tools, so no `before_tool`/`after_tool` events appear. In production use with tool invocations, each tool call generates an additional `before_tool`/`after_tool` pair, producing richer audit trails. The manuscript authoring workflow itself was conducted through Claude Code (the human-side interface), whose session transcripts are analyzed in Section S2; the sciClaw gateway was used for brief demonstration sessions captured here.

S7. Aggregate Summary

Category	Metric	Value
Interaction	Total session records	6,501
Interaction	Human messages	1,190
Interaction	Agent responses	1,872
Interaction	Tool invocations	1,108
Revision	Revision rounds	2

Category	Metric	Value
Revision	Tracked changes	7
Revision	Reviewer comments	10
Revision	Net words added	+1,886
References	References audited	15
References	Passed verification	11 (73.3%)
References	Failed verification	4 (26.7%)
References	Critical errors (hallucinated author)	1
Development	Git commits	16
Development	Duration	20.1 hours
Governance	Hook events captured	16
Governance	Metadata fields per event	8

S8. Limitations

1. **Interface separation.** The manuscript was authored using Claude Code as the human-side interface. The Claude Code session transcripts (6,501 records) provide rich interaction data, but they are Claude Code artifacts, not sciClaw gateway logs. The sciClaw governance hooks captured only 2 demonstration turns (16 events). This reflects a practical reality: during early development, the human researcher used Claude Code for its mature editing capabilities while sciClaw's gateway interface was still being built. Future case studies conducted entirely through the sciClaw gateway will produce richer governance audit trails.

2. **Stochastic outputs.** LLM outputs are non-deterministic. Re-running the same manuscript authoring workflow would produce different text, different tool call sequences, and potentially different results. The data presented here characterize one specific execution of the workflow, not a reproducible experiment in the traditional sense. What *is* reproducible is the audit trail itself: the analysis scripts in this directory will produce identical tables and figures from the same input data.

3. **Single case study (N=1).** This is one manuscript authored by one researcher with one agent configuration. Generalization to other researchers, topics, or agent configurations requires additional case studies.

4. **No cost data.** API token usage and associated costs were not systematically tracked during the authoring sessions. Future versions of the governance layer should capture token counts per LLM call to enable cost analysis.

S9. Reproduction Instructions

All analysis scripts are standard Python 3 with no required dependencies beyond the standard library (matplotlib is optional, for figure generation only). To reproduce:

```
cd manuscript/supplemental/
python3 01_session_analysis.py
python3 02_tracked_changes_analysis.py
```

```
python3 03_reference_audit_analysis.py
python3 04_git_history_analysis.py
python3 05_governance_analysis.py
python3 06_generate_tables.py  # requires matplotlib for figures
```

Session transcript paths are hardcoded to the author's local Claude Code project directory. To reproduce on a different machine, update the `SESSIONS_DIR` and `SESSION_FILES` paths in `01_session_analysis.py`.

Data Files

Source data (input):

- edits.json – Round 1 tracked changes (JSON)
- edits-round2.json – Round 2 tracked changes (JSON)
- reference-audit.md – Reference verification report (Markdown)
- hook-events.jsonl – Governance audit events (JSONL)

Computed results (output):

- session_analysis_results.json – Session metrics
- tracked_changes_results.json – Revision metrics
- reference_audit_results.json – Audit metrics
- git_history_results.json – Git metrics
- governance_results.json – Governance metrics
- aggregate_summary.json – Aggregate summary

Figures:

- figures/fig_s1_tool_usage.png – Tool usage distribution
- figures/fig_s2_commit_timeline.png – Commit activity timeline