NL2SciVis: A Benchmark for Natural Language to Scientific Visualization

Manish Mathai[1], Mengjiao Han[2], Janet Knowles[2], Victor A. Mateevitsi[2,3], Silvio Rizzi[2], and Hank Childs[1]

[1]University of Oregon, Eugene, Oregon, USA
[2]Argonne National Laboratory, Lemont, Illinois, USA
[3]University of Illinois Chicago, Chicago, Illinois, USA

Abstract

Scientific visualization (SciVis) tools enable exploration of complex scientific datasets but present a steep learning curve, creating hurdles for scientists by demanding technical expertise beyond their domain. AI agents powered by Large Language Models (LLMs) have the potential to address this issue, and yet their effectiveness and reliability in doing so are currently unclear. To address this gap, we introduce the NL2SciVis benchmark suite, which focuses on generating atomic scientific visualization operations in ParaView via Python code. To demonstrate its utility, we evaluate popular LLMs (Claude Sonnet 4.5, GPT-5, GPT-4o, and GPT-OSS-120B) on the suite under baseline, prompt modification, and scaffolding conditions, checking execution and pipeline validity across over 14,000 trials. Our evaluation shows that Sonnet 4.5 achieves a 96% pass rate while three GPT-family models range from 48–64%, with 92% of failures occurring during code execution rather than task misunderstanding.

CCS Concepts

• **Human-centered computing** → **Visualization systems and tools;** *Scientific visualization;* • **Computing methodologies** → *Natural language processing;*

1. Introduction

Visualization experts often guide domain scientists in using richly featured scientific visualization tools, such as ParaView [AGL05] and VisIt [C*12]. While these collaborations between domain scientists and visualization experts can yield many benefits, they can also represent a significant time investment. Large-language models (LLMs) could reduce this commitment by offering domain scientists advice in place of a human expert. However, it is unclear whether LLMs provide useful guidance. Recent work [LMB25] notes that such tools are underutilized; LLM assistance could also help scientists without ready access to experts.

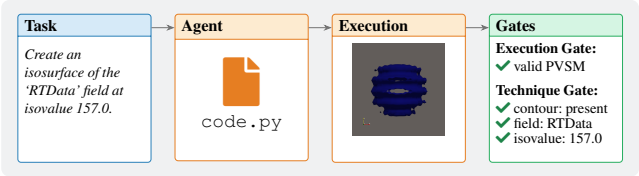
This work provides a benchmark suite to evaluate LLM efficacy for scientific visualization tasks using ParaView. It targets 3D scientific field data on structured or unstructured grids, unlike 2D information visualization frameworks like D3 [BOH11] or Vega [SRHH16]. Recent LLM-driven SciVis work [MYLP24, PMY*25, LMB25] also targets ParaView, placing our benchmark alongside that line of research. Defining the benchmark suite is challenging for two main reasons. First, LLMs provide nondeterministic results, so a single experiment is insufficient. Second, domain scientists use visualization tools to perform highly varied tasks. Running N trials across M tests for statistical rigor may be prohibitive. We address this issue by limiting the scope of the suite to basic operations. One motivation for this plan is that basic operations are the prerequisite for complex ones: an LLM that cannot handle basics cannot handle more complex tasks. That said, a benchmark suite is most useful when it informs the boundary between effective and ineffective. Fortunately, our experiments show that LLMs have success with some operations in our suite and less success with others. As LLMs improve and pass rates climb, this suite will become less useful. We discuss planned extensions in future work.

Our contribution is the NL2SciVis benchmark suite (Figure 1), which defines: 15 atomic operations across three categories (setup, adaptation, reporting) with evaluation checking both code execution and visualization correctness. We demonstrate its utility through three campaigns (baseline, prompt modifications, scaffolding) comprising over 14,000 trials. We observe that pass rates vary widely across models (48%–96%), with code execution as the primary failure mode, not task misunderstanding.

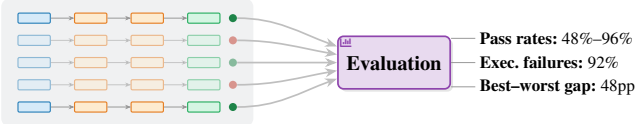
2. Related Work

NL2Vis benchmarks and systems. Early works produced natural language interfaces that parse text into chart specifications over tabular data [SBT*16, GDA*15, NSS21, YS20]. Neural approaches replaced hand-crafted parsing rules [LTL*22], and LLM-based code generation shifted the field toward executable scripts pro-

DOI: 10.2312/evs.20261017



(a) *Single trial.*



(b) *Benchmark.*

Figure 1: *NL2SciVis evaluation pipeline. (a)* Suite: *each trial feeds a task prompt to an LLM, executes the generated code in ParaView, and applies two validation gates. (b)* Benchmark: *30 trials × 15 tasks × 4 models yield pass rates from 48% to 96%, with 92% of failures at execution.*

duced by multi-step agentic pipelines [Dib23, SLY*25, YZW*24, CCA*25, BIS*26]. Standardized benchmarks drove this progress: nvBench synthesizes NL-visualization pairs from NL–SQL benchmarks [LTL*21], VisEval introduces execution-based evaluation [CZX*25], and Text2Vis adds multimodal outputs [RLJH25]. These systems and benchmarks operate exclusively on information visualization (2D charts from tabular data via Vega-Lite or Matplotlib), leaving scientific visualization without an equivalent benchmark.

LLM-driven scientific visualization. Recent systems extend LLM-driven visualization to scientific data, targeting flow fields [HXHT22, LZT25], volume rendering [ATW26], cosmological ensembles [TGB*25], and general ParaView/VTK pipelines [MYLP24,PMY*25,LMB25,BTR*26]. The closest work to our own is ChatVis [PMY*25], which pairs retrieval-augmented generation with iterative error correction to generate ParaView Python scripts and evaluates on 20 tasks using image similarity metrics. A direct comparison is not feasible because ChatVis evaluates end-to-end RAG pipelines with perceptual similarity, whereas NL2SciVis isolates atomic operations with deterministic state-file validation; the two benchmarks measure fundamentally different capabilities. ParaView-MCP [LMB25] takes a different approach, providing direct tool-use control of ParaView through a curated API, but validates through case studies and domain expert feedback rather than systematic benchmarks. Emerging frameworks extract agentic design patterns from visualization systems [DWL*25] and propose evaluation-centric paradigms for SciVis agents [AML*25]. Each system, however, evaluates through its own protocol, making cross-system comparison impossible and leaving open questions about which specific capabilities succeed or fail.

Code generation evaluation. General code benchmarks such as HumanEval [C*21] and MBPP [AON*21] test standalone function synthesis, while SWE-Bench [JYW*24] targets repository-level issue resolution; all assume standard library APIs. SciCode [T*24] targets scientific computing but not visualization, and AgentBench [LYZ*24] evaluates tool-using agents in generic environments without domain-specific stateful APIs.

Table 1: *The 15 atomic operations organized by workflow stage.*

Category	Operations
Setup (7)	Load, Contour, Slice, Volume Render, Streamline, Glyph, Threshold
Adaptation (4)	Camera, Colormap, Opacity, Representation
Reporting (4)	Mean, Min/Max, Area, Probe

To our knowledge, no benchmark targets NL-driven scientific visualization with deterministic validation, isolated capability testing, and cross-model comparison under controlled conditions. NL2SciVis addresses this gap.

3. NL2SciVis Suite

Scientific visualization workflows involve loading data, applying filters, adjusting parameters, and extracting results. Rather than assess full multi-step workflows, we decompose them into atomic operations to pinpoint what agents can and cannot do. Given a natural language instruction, the agent generates Python code targeting the ParaView API. Visualization operations are validated through ParaView state files (PVSM); reporting operations are validated against expected numerical ranges.

Operations and Validation: We define 15 atomic operations across three categories that follow a typical visualization workflow: pipeline setup, visual adaptation, and quantitative reporting (Table 1). The operations cover common visualization functions and are designed to verify their correctness deterministically.

Each operation satisfies four constraints: single intent, strict preconditions, independent execution, and deterministic defaults (Figure 2). Multi-step compositions, derived field construction, and qualitative adaptation goals (e.g., "make the contour more prominent") are excluded because they break one or more of these constraints. Future versions may relax them.

Each task is defined by a natural language prompt and corresponding validation checks. We validate against PVSM state files rather than rendered images as they provide direct structural access to the pipeline: filter types, field assignments, and parameter values can be verified exactly. Consider the contouring task as an example. The agent receives the prompt: *"Create an isosurface of the 'RTData' field at isovalue 157.0."* Validation happens in two stages: first, we verify that a valid PVSM file was produced; second, we confirm that (1) a Contour filter exists in the pipeline, (2) it operates on the correct field, and (3) the isovalue matches the requested value. The 5% tolerance was a default intended to admit almost-correct code, but in practice all contour trials matched the requested value exactly. Reporting tasks use a different validation approach. Because these operations produce numerical output rather than pipeline state, we verify that the printed result falls within an expected range. Supplement Section 1.5 presents per-task tolerances and an empirical drift analysis.

Evaluation Protocol: Two sequential gates deterministically assess each trial: **Execution Gate** checks whether the agent produced a valid state file; failures include syntax errors, runtime exceptions,

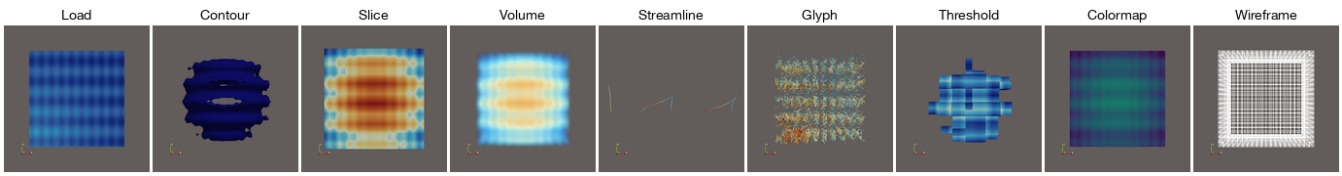


Figure 2: *Ground-truth renders for 9 of the 15 tasks; the remaining tasks produce non-visual outputs.*

or missing artifacts. **Technique Gate** checks whether the visualization method, field, and parameters are correct. A task passes only if both gates pass. The protocol verifies pipeline correctness, not visual quality. Distinguishing near-misses from total failures would require perceptual similarity metrics, which we leave to future work.

Our **primary metric** is pass rate: the proportion of trials where both execution and technique gates pass. We report 95% Wilson score confidence intervals, which provide better coverage than Wald intervals for proportions near 0 or 1 – relevant since some model-task combinations achieve 0% or 100% pass rates. For pairwise comparisons, we compute Cohen's h as the **effect size** to quantify practical significance independent of sample size. Following standard conventions: $|h| < 0.2$ is negligible, 0.2–0.5 is small, 0.5–0.8 is medium, and > 0.8 is large. For **significance testing**, we use chi-squared tests for independence to assess whether observed differences exceed chance variation. For multiple comparisons across 15 tasks, we apply Bonferroni correction ($\alpha = 0.05/15$) and report both raw and corrected p-values. As **secondary metrics**, we record lines of code generated, output token count, and execution latency for each trial to characterize generation cost and enable efficiency comparisons across models (see Supplement Section 5).

Dataset: The benchmark uses a synthetic wavelet dataset with scalar and gradient-derived vector fields suitable for all 15 operations. The dataset comes from VTK's `vtkRTAnalyticSource`, a deterministic source that produces a wavelet-shaped `RTData` scalar field, instantiated as a $40\times40\times40$ grid. We add a vector field computed from the scalar field's gradient, with a localized swirl term for non-zero vorticity. This synthetic choice ensures reproducibility but may limit generalization. Future versions may add public scientific datasets to test broader coverage.

Visualization tool: The operation taxonomy is tool-neutral. The evaluation protocol is ParaView-specific but portable to other scriptable applications (e.g., VisIt) by adapting the evaluator and runtime.

Model Selection: The benchmark evaluates any agent that accepts natural language and produces code. We demonstrate this across four models from three providers, representing different capability tiers and accessibility levels: Sonnet 4.5 (Anthropic), GPT-5 and GPT-4o (OpenAI), and GPT-OSS-120B, an open-weight model hosted on Argonne Leadership Computing Facility infrastructure [TCG*25]. All models receive identical system prompts instructing them to generate a self-contained Python function that performs the requested operation. System prompts and sampling parameters are given in Supplement Section 1.1.

Table 2: *Pass rates by model and operation category. N = 450 per model (30 × 15 tasks); 95% Wilson CIs.*

Model	Overall	Setup	Adapt.	Report.
Sonnet 4.5	96.0%	95.7%	92.5%	100.0%
GPT-5	63.6%	68.1%	90.8%	28.3%
GPT-4o	57.8%	55.7%	71.7%	47.5%
GPT-OSS-120B	48.4%	48.1%	83.3%	14.2%

4. Benchmark Demonstration

To illustrate the utility of the benchmark suite, we performed three campaigns with over 14,000 trials: Baseline (LLM performance on the 15 operations), Prompt Enhancement (effects of prompt modifications), and Scaffolding (effects of code templates). Supplement Section 1.3 details each campaign's construction.

Baseline Campaign: The baseline campaign establishes fundamental performance levels for each model. Each model executes all 15 tasks with a minimal system prompt containing only format instructions. We collected 30 runs per task (fixed prompt) per model, yielding 450 trials per model and 1,800 total baseline trials. Cross-run variation reflects model stochasticity at temperature 0.2. Table 2 presents pass rates by model and operation category. Sonnet 4.5 leads at 96.0%, a 32pp gap over the next model that is notably larger than typically reported on general code-generation benchmarks [C*21], suggesting scientific visualization poses distinct challenges. GPT-4o and GPT-5 show comparable performance (see Supplement Section 2.1).

Performance by Category and Operation: Breaking down by operation category (Table 2), two patterns emerge: **reporting operations** are the most difficult, requiring data-extraction APIs that differ from filter-creation patterns and are likely underrepresented in training data, while **adaptation operations** are easiest, involving simpler API calls with fewer dependencies. Sonnet 4.5 is the exception, showing consistently high performance across all categories.

Since each task uses a single prompt phrasing, individual operation results should be interpreted with caution. Operation-level results show binary outcomes: `Slice` achieves 100% across all models, while `Glyph` and `Threshold` see 0% for GPT-4o despite other models succeeding—yet GPT-4o reaches 100% on `Streamline`, suggesting model-specific API familiarity rather than pure task difficulty drives results. The most discriminating operation is `Probe` (point query), where non-Sonnet models fall below 20% while Sonnet achieves 100%. This operation chains location queries with data extraction, a multi-step pattern that frequently causes failures. The `Area` operation (surface area calculation), re-

quiring an uncommon filter combined with data extraction, shows a similar divide between Sonnet and the rest. Within setup operations, `Streamline` shows the widest performance spread: GPT-4o achieves 100% while Sonnet drops to 70%, its lowest for any operation. Per-model category rates and the complete operation-by-model matrix appear in Supplement Sections 2.2 and 2.3.

Prompt Enhancement Campaign: The prompt enhancement campaign tests whether specific prompt modifications affect pass rates. We append one of four conditions to the baseline prompt: (1) *Documentation*: "Please consult the ParaView v5.13 Python API documentation, when needed" (testing whether explicit documentation references help); (2) *Job stakes*: "Your performance on this task will directly affect my job offer" (testing emotional manipulation effects); (3) *Threat stakes*: "Your performance is time-sensitive and affects an urgent situation" (testing urgency framing); (4) *Negative reward*: "You start with $100 and lose it all if wrong" (testing loss aversion framing). We include these conditions because prior work has shown that prompt framing can significantly affect LLM behavior [LWZ*24], though this has not been systematically studied for code generation tasks.

For GPT models, all conditions produce negligible effects ($|h| < 0.1$) across prompt variations. Sonnet 4.5 shows a different pattern: three of four conditions *decrease* performance, with the documentation reference causing the largest drop (13.6pp, $h = -0.46$). The practical implication is that prompt engineering for high-performing models requires careful validation, as intuitive additions can backfire. Supplement Section 3 reports the full per-condition table.

Scaffolding Campaign: The scaffolding campaign provides a minimal code template showing the expected function format, filter creation pattern, and state-saving call. This campaign includes a concurrent control group to avoid cross-batch confounds. Figure 3 shows that GPT-5 benefits most from scaffolding (+9.7pp, $h = +0.20$), while GPT-4o and GPT-OSS-120B show negligible effects ($|h| \leq 0.01$) and Sonnet shows a small positive trend (+3.6pp, $h = +0.12$). Scaffolding helps models with partial task knowledge: GPT-5 understands the operations but benefits from structural guidance, whereas models that already know the patterns (Sonnet) or cannot use them (GPT-OSS) see little change. Supplement Section 4 reports the guideline-only condition.

Error Analysis: Of 5,177 total failures across all campaigns, 92.2% occur during code execution, 6.4% produce incorrect results from otherwise runnable code, and 1.4% fail to generate extractable code at all, indicating that the primary bottleneck is code generation, not task understanding.

5. Discussion and Conclusion

We introduced NL2SciVis, a benchmark of 15 atomic operations for evaluating LLM-driven scientific visualization, and demonstrated it across four models in over 14,000 trials. The dominance of execution-time failures points to code generation as the primary bottleneck, not task comprehension. This gap likely stems from the ParaView API's scarcity in public training corpora, compounded

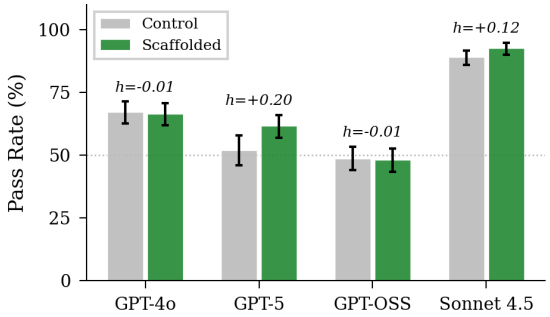


Figure 3: *Effect of code scaffolding on pass rates. Gray bars show within-experiment control; green bars show scaffolded condition. N = 450 per condition; error bars show 95% Wilson CIs; Cohen's h above.*

by stateful pipeline dependencies that syntax checking alone cannot detect (see Supplement Section 5). Model selection matters: the 32-percentage-point gap between the best and next-best models exceeds typical differences on general code-generation benchmarks, suggesting domain-API coverage in training data varies widely across model families. Prompt modifications show asymmetric effects across models (Supplement Section 3).

We scope evaluation to a single platform (ParaView), a synthetic wavelet dataset, and single-turn atomic operations. We leave cross-tool generalization to future work. The 96% pass rate of the strongest model signals that future iterations need harder operations to retain discriminative power. Extending NL2SciVis to real-world scientific datasets and multi-step composite tasks is a natural next step, as is periodic re-evaluation to track model progress. For reproducibility and community extension, the benchmark suite, evaluation harness, and all trial data are publicly available at this repository URL.

Acknowledgments: This research used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357. This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), for the project Argonne Base: Scientific Data Management and Visualization to Advance AI for Science, under Contract DE-AC02-06CH11357.

We used Claude Code and Grammarly for grammar correction, language polishing and rephrasing.

References

[AGL05] AHRENS J. P., GEVECI B., LAW C. C.: Paraview: An end-user tool for large-data visualization. In *The Visualization Handbook*, Hansen C. D., Johnson C. R., (Eds.). Academic Press / Elsevier, 2005, pp. 717–731. 1

[AML*25] AI K., MIAO H., LI Z., WANG C., LIU S.: An evaluation-centric paradigm for scientific visualization agents. In *VIS x GenAI 1st Workshop on GenAI, Agents, and the Future of VIS* (2025). 2

[AON*21] AUSTIN J., ODENA A., NYE M. I., BOSMA M., MICHALEWSKI H., DOHAN D., JIANG E., CAI C. J., TERRY M., LE Q. V., SUTTON C.: Program synthesis with large language models. arXiv:2108.07732. 2

[ATW26] AI K., TANG K., WANG C.: NLI4VolVis: Natural language interaction for volume visualization via LLM multi-agents and editable 3D gaussian splatting. *IEEE Trans. Vis. Comput. Graph. 32*, 1 (2026), 46–56. 2

[BIS*26] BROSSIER M., ISENBERG T., SCHONBORN K., UNGER J., ROMERO M., BJORKLUND J., YNNERMAN A., BESANCON L.: State of the art of LLM-enabled interaction with visualization, 2026. Preprint. 1

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. 1

[BTR*26] BISWAS A., TURTON T. L., RANASINGHE N. R., JONES S., LOVE B., JONES W., HAGBERG A., SHEN H., DEBARDELEBEN N., LAWRENCE E.: VizGenie: Toward self-refining, domain-aware workflows for next-generation scientific visualization. *IEEE Trans. Vis. Comput. Graph. 32*, 1 (2026), 1021–1031. 2

[C*12] CHILDS H., ET AL.: Visit. In *High Performance Visualization - Enabling Extreme-Scale Scientific Insight*, Bethel E. W., Childs H., Hansen C. D., (Eds.), Chapman and Hall / CRC computational science series. CRC Press, 2012. 1

[C*21] CHEN M., ET AL.: Evaluating large language models trained on code. arXiv:2107.03374. 2, 3

[CCA*25] CHEN Z., CHEN J., ARIK S. Ö., SRA M., PFISTER T., YOON J.: CoDA: Agentic systems for collaborative data visualization. arXiv:2510.03194. 1

[CZX*25] CHEN N., ZHANG Y., XU J., REN K., YANG Y.: Viseval: A benchmark for data visualization in the era of large language models. *IEEE Trans. Vis. Comput. Graph. 31*, 1 (2025), 1301–1311. 2

[Dib23] DIBIA V.: LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2023* (2023), Association for Computational Linguistics, pp. 113–126. 1

[DWL*25] DHANOA V., WOLTER A., LEÓN G. M., SCHULZ H., ELMQVIST N.: Agentic visualization: Extracting agent-based design patterns from visualization systems. *IEEE Computer Graphics and Applications 45*, 6 (2025), 89–100. 2

[GDA*15] GAO T., DONTCHEVA M., ADAR E., LIU Z., KARAHALIOS K. G.: Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST 2015* (2015), ACM, pp. 489–500. 1

[HXHT22] HUANG J., XI Y., HU J., TAO J.: FlowNL: Asking the flow data in natural languages. *IEEE Trans. Vis. Comput. Graph.* (2022), 1–11. 2

[JYW*24] JIMENEZ C. E., YANG J., WETTIG A., YAO S., PEI K., PRESS O., NARASIMHAN K. R.: Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024* (2024), OpenReview.net. 2

[LMB25] LIU S., MIAO H., BREMER P.: ParaView-MCP: An autonomous visualization agent with direct tool use. In *2025 IEEE Visualization and Visual Analytics (VIS)* (2025), pp. 61–65. 1, 2

[LTL*21] LUO Y., TANG N., LI G., CHAI C., LI W., QIN X.: Synthesizing natural language to visualization (NL2VIS) benchmarks from NL2SQL benchmarks. In *SIGMOD '21: International Conference on Management of Data* (2021), ACM, pp. 1235–1247. 2

[LTL*22] LUO Y., TANG N., LI G., TANG J., CHAI C., QIN X.: Natural language to visualization by neural machine translation. *IEEE Trans. Vis. Comput. Graph. 28*, 1 (2022), 217–226. 1

[LWZ*24] LI C., WANG J., ZHANG Y., ZHU K., WANG X., HOU W., LIAN J., LUO F., YANG Q., XIE X.: The good, the bad, and why: Unveiling emotions in generative AI. In *Forty-first International Conference on Machine Learning, ICML 2024* (2024), vol. 235 of *Proceedings of Machine Learning Research*, PMLR, pp. 28905–28934. 4

[LYZ*24] LIU X., YU H., ZHANG H., XU Y., LEI X., LAI H., GU Y., DING H., MEN K., YANG K., ZHANG S., DENG X., ZENG A., DU Z., ZHANG C., SHEN S., ZHANG T., SU Y., SUN H., HUANG M., DONG Y., TANG J.: AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024* (2024), OpenReview.net. 2

[LZT25] LI Z., ZHANG W., TAO J.: FlowLLM: Large language model driven flow visualization. *Visual Informatics 9*, 3 (2025), 100241. 2

[MYLP24] MALLICK T., YILDIZ O., LENZ D., PETERKA T.: ChatVis: Automating scientific visualization with a large language model. In *Proceedings of the SC '24 Workshops, SC-W '24* (2024), IEEE Press, pp. 49–55. 1, 2

[NSS21] NARECHANIA A., SRINIVASAN A., STASKO J. T.: NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Trans. Vis. Comput. Graph. 27*, 2 (2021), 369–379. 1

[PMY*25] PETERKA T., MALLICK T., YILDIZ O., LENZ D., QUAMMEN C., GEVECI B.: ChatVis: Large language model agent for generating scientific visualizations. In *2025 IEEE 15th Symposium on Large Data Analysis and Visualization (LDAV)* (2025), pp. 22–32. 1, 2

[RLJH25] RAHMAN M., LASKAR M. T. R., JOTY S., HOQUE E.: Text2Vis: A challenging and diverse benchmark for generating multimodal visualizations from text. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025* (2025), Association for Computational Linguistics, pp. 31849–31874. 2

[SBT*16] SETLUR V., BATTERSBY S. E., TORY M., GOSSWEILER R., CHANG A. X.: Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology, UIST 2016* (2016), ACM, pp. 365–377. 1

[SLY*25] SHUAI Z., LI B., YAN S., LUO Y., YANG W.: DeepVIS: Bridging natural language and data visualization through step-wise reasoning. *IEEE Trans. Vis. Comput. Graph.* (2025), 1–11. 1

[SRHH16] SATYANARAYAN A., RUSSELL R., HOFFSWELL J., HEER J.: Reactive Vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 659–668. 1

[T*24] TIAN M., ET AL.: SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems 38, NeurIPS 2024* (2024). 2

[TCG*25] TANIKANTI A., CÔTÉ B., GUO Y., CHEN L., SAINT N., CHARD R., RAFFENETTI K., THAKUR R., URAM T., FOSTER I., PAPKA M. E., VISHWANATH V.: FIRST: Federated inference resource scheduling toolkit for scientific AI model access. In *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2025). 3

[TGB*25] TAM J. Z., GROSSET P., BANESH D., RAMACHANDRA N., TURTON T. L., AHRENS J. P.: InferA: A smart assistant for cosmological ensemble data. In *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2025), ACM. 2

[YS20] YU B., SILVA C. T.: Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE Trans. Vis. Comput. Graph. 26*, 1 (2020), 1–11. 1

[YZW*24] YANG Z., ZHOU Z., WANG S., CONG X., HAN X., YAN Y., LIU Z., TAN Z., LIU P., YU D., LIU Z., SHI X., SUN M.: MatPlotAgent: Method and evaluation for LLM-based agentic scientific data visualization. In *Findings of the Association for Computational Linguistics: ACL 2024* (2024), Association for Computational Linguistics, pp. 11789–11804. 1