Gardener: An Agentic AI System for Single-Cell RNA Sequence Analysis

Junhan Liu[1]* **Zhenke Liu**[1]* **Yongcheng Shi**[1] **Peilin Yu**[1]
Minxing Zhang[2] **Jiapeng Zhang**[3]†

[1]Brown University
[2]Duke University
[3]University of Southern California
{junhan_liu, zhenke_liu, yongcheng_shi, peilin_yu}@brown.edu,
minxing.zhang@duke.edu, jiapengz@usc.edu

Abstract

Gardener is an interactive agentic system for single-cell RNA-seq (scRNA-seq) analysis that enables expert-steered, iterative workflows under strict data-residency requirements. Existing large language model (LLM)-based analysis agents commonly encode workflow progress as implicit conversational state and rely on cloud-centric execution, which hinders traceability and auditability and complicates keeping sensitive expression data on-device. Gardener grounds cloud-side reasoning in a local, on-device scientific engine and an Experiment Management Kernel (EMK) that externalizes analysis progress as persistent, immutable snapshots linked by lineage. This explicit state representation supports rollback, branching, and comparison of alternative analysis paths while reusing prior computation. Gardener enforces data isolation by design: cloud-hosted LLMs operate only on snapshot identifiers and sanitized summaries, while raw expression matrices and local artifacts remain on the user's device. A local graphical user interface (GUI) provides human-in-the-loop steering and inspection of workflow state and outputs. Gardener is released as an open-source desktop application for macOS and Windows under the Apache License 2.0.

1 Introduction

Recent advances in biomedical large language models (LLMs) (Bolton et al., 2024; Luo et al., 2022) and agent frameworks (Huang et al., 2025; Xiao et al., 2024; Mao et al., 2025; Gao et al., 2025) have enabled natural-language interfaces for scRNA-seq analysis pipelines built on scientific toolkits (Wolf et al., 2018; Domínguez Conde et al., 2022; Fang et al., 2023). However, most existing systems are cloud-centric and implicitly treat the LLM as the primary workspace for memory and state tracking. This design is misaligned with real-world scRNA-seq analysis, which is iterative, stateful, and error-prone. Experts must inspect intermediate results, tune hyperparameters, and compare alternatives before accepting biological conclusions. Moreover, these systems operate in a multi-party setting involving users and cloud-hosted LLMs, imposing a data-residency constraint: sensitive data must remain on-device even when cloud models are used for reasoning and planning.

To address these mismatches and lower the interaction barrier to scRNA-seq analysis while preserving expert oversight for bioinformaticians, wet-lab biologists, and trainees, we present **Gardener**[1], an interactive agentic system for expert-steered, residency-preserving scRNA-seq analysis. Gardener employs a hybrid orchestration pattern that grounds cloud-hosted reasoning in a local scientific engine. Through a Resident State Protocol (RSP), high-level planning is tethered to the local environment, ensuring local data residency by exposing only contextual coordinates and sanitized summaries to the cloud. The underlying Experiment Management Kernel (EMK) externalizes analysis as immutable, versioned snapshots, enabling efficient rollback and branching. For expert steering, a local Graphical User Interface (GUI) provides real-time inspection of states and biological outputs.

Gardener is a prototype interactive agentic system for scRNA-seq analysis with snapshot-based version control for analysis states. We make the following system contributions:

- **Local Scientific Engine with an Experiment Management Kernel.** Gardener executes scRNA-seq workflows on-device by coupling an EMK with refactored scientific toolkits, enabling rollback, branching, and comparison during non-linear workflow exploration.

*Equal contribution
†Corresponding author: jiapengz@usc.edu

[1]https://github.com/CrossOmics/Gardener-Agent

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 408–417

- **Data residency through architectural isolation** Gardener decouples cloud-hosted reasoning from on-device execution by enforcing a Resident State Protocol that restricts cloud-facing interactions to contextual coordinates and sanitized summaries. This ensures that sensitive scRNA-seq artifacts remain under local jurisdiction, preventing raw data exposure to cloud environments by design.

- **Human-in-the-loop interaction.** Gardener provides a local GUI for expert–agent interaction, enabling users to steer, inspect, and revise the analysis workflow throughout iterative exploration.

2 System and Design

Gardener operates in a multi-party setting: execution and data stay on-device, while cloud-hosted LLMs provide reasoning and planning. This separation imposes a data-residency constraint—sensitive scRNA-seq artifacts (e.g., expression matrices) must remain local—yet the cloud model still requires enough context to coordinate the analysis.

To satisfy these requirements, Gardener separates *reasoning* from *execution*. Cloud-hosted LLMs handle planning, while all computation and artifacts remain on-device. This separation is realized by two components: a *Local Scientific Engine* (§2.1) and a cloud-facing *Resident State Protocol* (RSP) (§2.2). The Local Scientific Engine couples an Experiment Management Kernel (EMK) with refactored scRNA-seq toolkits (Mukherjee et al., 2025; Wolf et al., 2018; Domínguez Conde et al., 2022; Fang et al., 2023) that operate over EMK states. The RSP restricts cloud interactions to snapshot references and compact summaries, ensuring raw data never leaves the device. Building on these foundations, Gardener adopts a modular agent architecture with Perception-and-Planning, Tool, Memory, and Environment modules (§2.3); only the **Perception-and-Planning Module** communicates with cloud-hosted LLMs.

2.1 Local Scientific Engine

The EMK is built around the `AnalysisSnapshot` abstraction (Figure 1), the core record for versioned analysis state. Each step commits a new immutable snapshot (`snapshot_id`); metadata is stored in a local database, while large artifacts remain on-device and are referenced by `snapshot_path`. Lineage links support rollback and branching over versioned analysis states, enabling iterative revision and side-by-side comparison of alternative configurations. Each snapshot also stores a `dataset_id` that anchors it to the underlying dataset for the addressing of the state on the device. Refactored toolkits operate on EMK snapshots via a functional state-transition pattern: they load a source snapshot, execute computation, and commit a new snapshot as output.

```python
class AnalysisSnapshot(BaseModel):
    snapshot_id: str # unique, indexed
    parent_snapshot_id: str | None
    dataset_id: str # foreign key to Dataset
    branch_name: str
    snapshot_path: str
    params_json: dict | None
    create_time: datetime
```

Figure 1: Core snapshot record in the EMK. Each step commits an immutable snapshot with lineage (`parent_snapshot_id`) and branch identity (`branch_name`); large artifacts remain on-device and are referenced via `snapshot_path`.

```python
def resolve_coordinate(snapshot:
    AnalysisSnapshot) -> tuple[str, str,
    str]:
    dataset =
        Dataset.get(snapshot.dataset_id)
    project_id =
        dataset.project_id.project_id
    return (project_id,
        snapshot.dataset_id,
        snapshot.snapshot_id)
```

Figure 2: Resolving the contextual coordinate (`project_id`, `dataset_id`, `snapshot_id`) from a snapshot record via local metadata relationships.

Gardener uses the *contextual coordinate* (`project_id`, `dataset_id`, `snapshot_id`) to address on-device analysis state and artifacts; the full coordinate is resolved from `snapshot_id` through local metadata relationships.

2.2 Resident State Protocol

To enforce the separation between on-device execution and cloud-hosted LLM reasoning (Figure 3), we define a *Resident State Protocol* consisting of a coordinate-only request contract and a sanitized-summary response contract. Under this protocol, the cloud-facing planner operates on contextual coordinates and compact summaries derived from lo-

cal state, rather than raw expression matrices or local file-system pointers. Figure 3 shows an illustrative implementation sketch of the request/response interface and sanitization step, and Appendix D provides a real example.

```
class AnalysisRequest(BaseModel):
    project_id: str
    dataset_id: str
    snapshot_id: str
EXCLUSION_SET = {"plot", "path", "json",
    "csv"}
def sanitize_for_agent(raw_metadata: dict)
    -> dict:
    return {
        k: v for k, v in
            raw_metadata.items()
        if not any(kw in k.lower() for kw
            in EXCLUSION_SET)
    }
class SnapshotSummary(BaseModel):
    snapshot_id: str
    input_params: dict
    distilled_output: dict
```

Figure 3: Illustrative sketch for Gardener's Resident State Protocol. Requests reference analysis state via contextual coordinates, and responses expose sanitized summaries for cloud-side reasoning.

2.3 Agent Modules

Perception-and-Planning Module. This module is the only component that communicates with cloud-hosted LLMs. Perception assembles the planning context from the current user message, relevant interaction history from the **Memory Module**, project context, and recent execution results, while exposing only protocol-compliant information (contextual coordinates and sanitized summaries; §2.2). Planning then produces either a direct response, a tool-backed execution request, or a request for additional user input. The execution workflow and control loop are described in §3.2.

Tool Module. This module encapsulates domain-specific toolkits as atomic, stateless operators with semantic schemas exposed to the **Perception-and-Planning Module**. It therefore serves as a semantic tool layer that lets the planner invoke local analysis capabilities through natural-language-driven execution plans, rather than direct access to toolkit internals. Tools consume contextual coordinates and execution parameters, delegate computation to the Local Scientific Engine (§2.1), and return a resulting `snapshot_id` with sanitized summaries for downstream reasoning.

Memory Module. Unlike the EMK, which stores versioned analysis state, the **Memory Module** stores the agent's interaction history as a structured log of typed events (e.g., user inputs, tool calls/results, and agent outputs) with associated payloads. This log supports context restoration and summarization across turns, allowing the agent to recover prior decisions and resume iterative workflows with traceable rationale.

Environment Module. The Environment Module is the on-device analysis environment: datasets, versioned snapshots, and their derived artifacts, addressed by contextual coordinates. It is observed through sanitized summaries and mutated only through tool-triggered snapshot commits executed by the **Tool Module**; the **Perception-and-Planning Module** has no direct access to raw data or local artifacts. Gardener provides a local GUI as the human-in-the-loop surface for steering and inspection (§3).

3 Interactive Workflow Orchestration

Building on the module design in Section §2.3, this section describes Gardener's interactive GUI§3.1 and workflow orchestration§3.2.

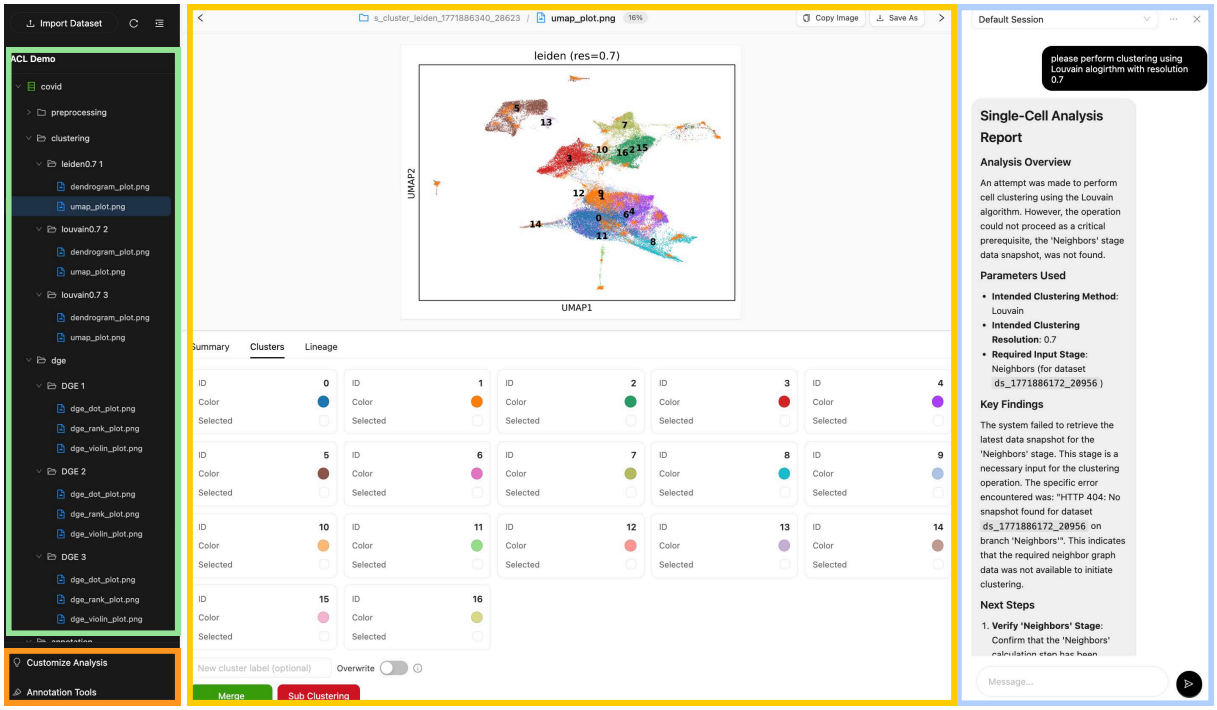
3.1 GUI for Expert-Steered Orchestration

The local GUI (Figure 4a) serves as a **HITL** control plane that transforms Gardener from a black-box pipeline into a transparent, expert-steered system by bridging expert oversight with underlying modules.

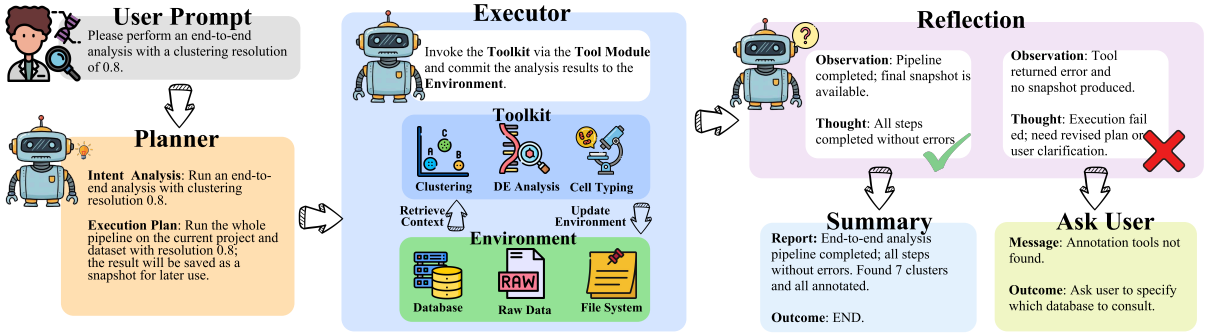
Specifically, the **Navigator** (green block) connects the expert with the **EMK** (§2.1) by externalizing analysis state as a navigable tree of `AnalysisSnapshots`, enabling high-level management such as lineage rollback or branching while the agent synchronizes contextual coordinates accordingly.

The **Analysis Inspector** (yellow block) bridges the expert with execution outputs by **visualizing analytical deliverables** (e.g., UMAP, DotPlots) associated with the current **active EMK snapshot** (§2.1), allowing for biological validation before permitting the agent to proceed with downstream reasoning.

The **Configuration Panel** (orange block) links the expert to the **Tool Module** (§2.3) for parameter customization and prior injection, enabling direct



(a) GUI overview of the HITL control plane: the Navigator (green) exposes the snapshot lineage for rollback/branching, the Analysis Inspector (yellow) visualizes biological outputs for validation, the Configuration Panel (orange) supports tool customization, and the Chat Panel (blue) provides natural-language interaction with the agent.



(b) A user prompt is processed by the planner (intent analysis and execution planning), followed by the executor (tool invocation and update the environment) and reflection (outcome evaluation). Based on the reflection result, the agent either returns a final report, requests user clarification, or replans when the failure is recoverable.

steering such as refining clusters or sub-clustering specific populations based on real-time inspection.

Finally, the **Chat Panel** (blue block) bridges the expert with the **Perception-and-Planning Module** (§2.3), facilitating natural-language interaction to refine agent intent and provide strategic guidance throughout the iterative workflow. Thus, the GUI ensures every agent action is auditable and reversible, fulfilling the requirements of a truly interactive agentic system with guaranteed data residency.

3.2 Workflow Orchestration

Figure 4b shows the execution-time control loop, implemented as a graphical state machine over a shared execution state containing dialogue `messages`, contextual coordinates, the current plan, execution results, and reflection feedback. Each turn begins at the `planner node`, which interprets the interaction state and either responds directly or issues a tool-backed execution request to the `executor node`. The `executor node` invokes local toolkits through the Local Scientific Engine, resulting in environment updates. The `reflection node` evaluates the outcome and either finalizes the turn with a structured summary, triggers replanning for recoverable failures, or routes to `ask_user_node` when additional user input is required.

Ann.	Meth.	Bio Fidelity		Robust.†	Eff.	LLM†		Likert (1–5)		
		Cov.	Missing	Succ.	Time	Tok.	$	Trans.	Steer.	Resid.
Biologist	Gardener	10/10	None	5/5	6m31s	15,540	$0.04	4.00	4.00	4.00
	Biomni	10/10	None	N/A	12m17s	N/A	N/A	4.00	5.00	2.00
Bioinformatician 1	Gardener	10/10	None	5/7	7m01s	13,789	$0.03	4.00	4.00	4.00
	Biomni	9/10	pDC	N/A	11m38s	N/A	N/A	4.00	4.00	2.00
Bioinformatician 2	Gardener	10/10	None	7/8	8m22s	18,852	$0.05	4.00	4.00	4.00
	Biomni	10/10	None	N/A	14m15s	N/A	N/A	3.00	4.00	2.00
Avg.	Gardener	10.00	None (3/3)	17/20	7m18s	16,060	$0.04	4.00	4.00	4.00
	Biomni	9.67	pDC (1/3)	N/A	12m43s	N/A	N/A	3.67	4.33	2.00

Table 1: **Per-annotator evaluation**. Cov.=coverage over 10 major classes after ontology mapping; Missing=absent class(es). Eff.=total wall-clock time for the user-defined workflow. †Tool-use success rate and token consumption are Gardener-specific; therefore, Biomni is reported as N/A for these fields.

4 Experiments and Evaluation

We report an expert-driven case study on a COVID-19 PBMC scRNA-seq dataset comprising 59,506 cells (Ballestar et al.). The study was conducted by three independent domain experts (one biology PhD candidate and two bioinformaticians) who were not involved in the system's development.

Task. The experts were tasked with reconstructing the dataset's major cell-type landscape by producing biologically plausible cell-type labels and verifying that rare populations were not overlooked during iterative, trial-and-error exploration.

Metrics. We evaluate the system along four dimensions:

- **Biological Fidelity**: recovery of the 10 ground-truth major cell types after ontology mapping (coverage and missing classes).
- **Expert Assessment**: 5-point Likert ratings (1=strongly disagree, 5=strongly agree) on (i) transparency, (ii) steerability, and (iii) data residency/trust. More details included in E.
- **Computational Efficiency**: end-to-end wall-clock time for the user-defined workflow.
- **Operational Robustness (Gardener-only)**: tool-use success rate, measured from Gardener's internal logs.

Because Biomni is provided as a cloud-based web service with inaccessible backend logs and a disparate hardware/software stack, internal execution metrics such as tool-use success and token consumption are not directly measurable for Biomni; accordingly, we compare against Biomni primarily on biological fidelity and end-to-end wall-clock efficiency, while reporting robustness, token usage, and Likert ratings for Gardener as system-internal and expert-assessment results.

Dataset. The dataset defines a 10-class cell-type ontology: *B cell*, *CD4 T cell*, *CD8 T cell*, *Dendritic Cell*, *Monocyte*, *NK cell*, *pDC*, *Plasmablast*, *Platelet*, and *T cell*. The experts map method-specific labels (from Gardener and Biomni) to this ontology and report coverage and missing classes.

4.1 Quantitative Results

Biological fidelity. Table 1 shows that Gardener achieves consistent major-class recovery across all annotators (10/10 in 3/3 runs). In contrast, Biomni misses the rare *pDC* population in 1/3 runs. Visualization is provided at C.

Operational robustness. Gardener exhibits high tool-use reliability across runs, achieving 5/5, 5/7, and 7/8 successful tool calls (17/20 overall). The lower success ratios in the latter two runs are attributable to user-input issues (e.g., a misspelling of *Leiden*) and an initial misunderstanding of the annotation tool, which the agent detected and corrected during execution.

End-to-end efficiency and LLM usage. For each workflow executed, Gardener completes in 7m18s on average, compared to 12m43s for Biomni (Table 1). Gardener also incurs modest LLM usage, averaging 16,060 tokens and approximately $0.04 per run with Gemini 2.5 Flash (Comanici et al., 2025). Notably, Bioinformatician 2 performed an additional verification by rerunning clustering with Louvain, which increased end-to-end time (8m22s), token consumption (18,852), and tool usage (7/8)

relative to the other Gardener runs. These results suggest that Gardener provides an efficient and economical path for making scRNA-seq analysis more accessible through iterative, expert-in-the-loop workflows.

4.2 Qualitative Findings

Rare-cell failure mode. Across annotators, the experts attributed Biomni's occasional omission of *pDC* to sensitivity in preprocessing, where default parameter choices can disproportionately affect rare populations. This aligns with their prior experience that rare cell types are especially vulnerable to configuration-dependent filtering.

Expert assessment (Likert). Gardener receives higher average scores on transparency (4.00 vs. 3.67) and data residency/trust (4.00 vs. 2.00), while Biomni is rated higher on steerability (4.33 vs. 4.00). Experts consistently preferred workflows in which raw expression matrices remain on the local device, and viewed data residency as a prerequisite to adopt LLM-assisted scRNA-seq analysis under typical biomedical governance constraints.

The biologist also valued Biomni's fully autonomous end-to-end workflow, describing it as convenient for users who prefer minimal manual intervention or are less familiar with the scRNA-seq analysis toolkit. In contrast, bioinformaticians preferred workflows that allow them to customize the analysis toolkit and inject domain priors, and considered such steerability important for iterative exploration.

5 Related Work

Domain-adapted biomedical LLMs improve biomedical language understanding and retrieval (Luo et al., 2022; Bolton et al., 2024; Rizvi et al., 2025; Levine et al., 2024; Li et al., 2024). In parallel, agent frameworks extend LLMs from single-turn generation to multi-step reasoning with tool use (e.g., retrieval and function calls) (Yao et al., 2023; Nakano et al., 2021; Schick et al., 2023; Patil et al., 2024). Building on these advances, recent systems connect LLM reasoning with bioinformatics tools for single-cell analysis (Xiao et al., 2024; Huang et al., 2025; Gao et al., 2025; Mao et al., 2025; Lu et al., 2024; Gao et al., 2024).

Despite promising results, most existing scRNA-seq agents treat analysis progress as implicit conversational context rather than a persistent, versioned state, which makes non-linear exploration difficult to support and hard to audit. Moreover, many prototypes assume cloud-centric execution or data access patterns that are incompatible with common data residency constraints in genomics.

Gardener differs by grounding LLM planning in an on-device Local Scientific Engine with an Experiment Management Kernel: each step is recorded as an immutable `AnalysisSnapshot` with lineage, and only derived summaries cross the privacy boundary to cloud-hosted LLMs for reasoning. This design supports human-in-the-loop steering while providing explicit state evolution for reproducible, privacy-preserving scRNA-seq workflows.

6 Discussion, Limitation, and Future Work

Gardener adopts a human-in-the-loop (HITL) design rather than full autonomy, reflecting the iterative and non-linear nature of scRNA-seq analysis. In practice, key decisions such as preprocessing choices, clustering resolution, marker interpretation, and rare-population validation require expert accountability. Gardener therefore positions the agent as a state-aware executor under expert supervision: the LLM planner translates user intent into local tool actions, while the user retains responsibility for scientific validation. The Experiment Management Kernel records each state transition as an immutable snapshot with lineage, enabling auditing, rollback, and branching during exploratory analysis.

A limitation of the current evaluation is its scope. Our expert-driven case study uses one COVID-19 PBMC (Ballestar et al.) dataset and three independent annotators. Although this setting reflects a realistic interactive workflow and covers major immune-cell annotation, it does not fully characterize Gardener's performance across the broader diversity of scRNA-seq studies, including non-immune tissues, stronger batch effects, higher sparsity, cross-platform variation, and different noise profiles. Future work will evaluate Gardener on a larger benchmark suite spanning multiple tissues, sequencing technologies, dataset scales, and annotation granularities. This broader evaluation will allow us to quantify robustness under more heterogeneous biological and technical conditions.

Another limitation is that we report end-to-end workflow time but do not separately isolate the

latency introduced by the Resident State Protocol and the Experiment Management Kernel. In Gardener, RSP sanitization, snapshot commitment, metadata indexing, and summary generation are necessary for data residency and traceability, but they may introduce overhead compared with a standard non-agentic local pipeline. Future work will include a component-level latency study that decomposes total runtime into scientific computation, RSP sanitization, EMK snapshot creation, metadata indexing, LLM planning, and GUI synchronization. This analysis will clarify the performance cost of residency-preserving orchestration and identify optimization targets.

Gardener's local-first design also introduces resource trade-offs. Since raw expression matrices and derived artifacts remain under user-controlled infrastructure, large scRNA-seq datasets may place substantial pressure on local RAM, CPU, and disk capacity when executed on a personal workstation. The immutable snapshot design further increases storage use because each analysis step commits a persistent state to support reproducibility, rollback, and branching. Future versions will support connecting the Local Scientific Engine to user-managed high-performance computing (HPC) environments or institutional compute clusters. Under this deployment mode, the GUI and cloud-facing planner would continue to operate through the same Resident State Protocol, while computation-heavy steps would be dispatched to trusted user-controlled compute resources rather than third-party cloud services. This extension would preserve the residency boundary while allowing Gardener to scale to larger datasets and more expensive workflows. We will also investigate incremental and content-addressed storage mechanisms, including artifact deduplication, sparse-delta snapshots, lazy materialization, and automatic compaction of inactive branches, to reduce disk growth while preserving auditability.

RSP intentionally limits the cloud planner to contextual coordinates and sanitized summaries. This restriction is central to Gardener's data-residency guarantee, but it may reduce the planner's access to subtle biological signals that are visible in raw matrices, plots, or detailed local artifacts. In the current design, this risk is mitigated by expert inspection through the local GUI, where biological outputs remain available for human validation. Future work will study richer privacy-preserving summaries that expose biologically useful aggregate information without leaking raw data or local file-system context.

Overall, Gardener should be viewed as a system-level contribution that combines local scientific execution, explicit snapshot-based state management, cloud-side planning, and GUI-based expert steering under a data-residency constraint. While individual components build on existing ideas in LLM agents, workflow management, and scRNA-seq toolkits, their integration around an enforceable residency boundary enables a practical interaction model for privacy-sensitive biomedical analysis. Future work will extend this foundation through broader evaluation, component-level overhead analysis, scalable snapshot storage, richer privacy-preserving summaries, and adjustable autonomy.

References

Esteban Ballestar, DL Farber, S Glover, B Horwitz, K Meyer, M Nikolić, J Ordovas-Montanes, P Sims, A Shalek, N Vandamme, and 1 others. Single cell profiling of covid-19 patients: an international data resource from multiple tissues." medrxiv (2020). *URL https://www. medrxiv. org/content/10.1101/2020.11*, 20:v1.

Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjou, Jonathan Frankle, Percy Liang, Michael Carbin, and 1 others. 2024. Biomedlm: A 2.7 b parameter language model trained on biomedical text. *arXiv preprint arXiv:2403.18421*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

C Domínguez Conde, Chao Xu, Louie B Jarvis, Daniel B Rainbow, Sara B Wells, Tamir Gomes, SK Howlett, O Suchanek, K Polanski, HW King, and 1 others. 2022. Cross-tissue immune cell analysis reveals tissue-specific features in humans. *Science*, 376(6594):eabl5197.

Zhuoqing Fang, Xinyuan Liu, and Gary Peltz. 2023. Gseapy: a comprehensive package for performing gene set enrichment analysis in python. *Bioinformatics*, 39(1):btac757.

Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. 2024. Empowering biomedical discovery with ai agents. *Cell*, 187(22):6125–6151.

Yiming Gao, Zhen Wang, Jefferson Chen, Mark Antkowiak, Mengzhou Hu, JungHo Kong, Dexter Pratt,

Jieyuan Liu, Enze Ma, Zhiting Hu, and Eric P. Xing. 2025. scpilot: Large language model reasoning toward automated single-cell analysis and discovery. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, and 1 others. 2025. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pages 2025–05.

Daniel Levine, Syed A Rizvi, Sacha Lévy, Nazreen Pallikkavaliyaveetil, David Zhang, Xingyu Chen, Sina Ghadermarzi, Ruiming Wu, Zihe Zheng, Ivan Vrkic, and 1 others. 2024. Cell2sentence: Teaching large language models the language of biology. In *International Conference on Machine Learning*, pages 27299–27325. PMLR.

Cong Li, Qingqing Long, Yuanchun Zhou, and Meng Xiao. 2024. screader: Prompting large language models to interpret scrna-seq data. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 665–672. IEEE.

Yen-Chun Lu, Ashley Varghese, Rahul Nahar, Hao Chen, Kunming Shao, Xiaoping Bao, and Can Li. 2024. scchat: A large language model-powered copilot for contextualized single-cell rna sequencing analysis. *Preprint*, bioRxiv:2024.10.01.616063.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409.

Yuren Mao, Yu Mi, Peigen Liu, Mengfei Zhang, Hanqing Liu, and Yunjun Gao. 2025. scagent: Universal single-cell annotation via a llm agent. *arXiv preprint arXiv:2504.04698*.

Chandra Sekhar Mukherjee, Joonyoung Bae, and Jiapeng Zhang. 2025. Corespect: Enhancing clustering algorithms via an interplay of density and geometry. *arXiv preprint arXiv:2507.08243*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Syed Asad Rizvi, Daniel Levine, Aakash Patel, Shiyang Zhang, Eric Wang, Curtis Jamison Perry, Nicole Mayerli Constante, Sizhuang He, David Zhang, Cerise Tang, and 1 others. 2025. Scaling large language models for next-generation single-cell analysis. *BioRxiv*, pages 2025–04.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. 2018. Scanpy: large-scale single-cell gene expression data analysis. *Genome biology*, 19(1):15.

Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, and 1 others. 2024. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

A Ethical Considerations

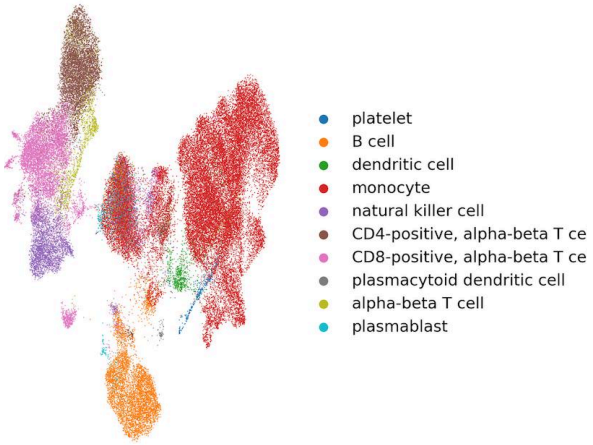
Gardener targets scRNA-seq workflows, where raw expression artifacts may be sensitive. To reduce exposure when using cloud-hosted LLMs, Gardener enforces the Resident State Protocol (RSP): cloud-side planning operates only on contextual coordinates and sanitized summaries, while raw artifacts and execution remain on-device.

To mitigate privacy and governance concerns, our evaluation uses a publicly available scRNA-seq dataset rather than proprietary or patient-identifiable data.

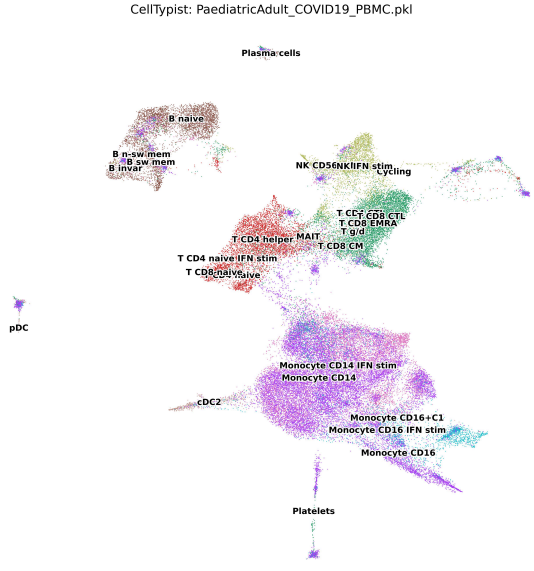
B Acknowledgments

We thank Jien Li for recruiting experts to conduct the evaluation.

C Supplementary Material



(a) Biomni Annotation.



(b) Gardener Annotation.

We provide visualization of the final annotations from both systems. Cell-type labels are mapped to the shared 10-class ontology described in Section 4. The rare pDC population is highlighted in Figure 5b.

D Data Residency

```
planner_plan:
  tool_name: "run_clustering"
  suggestion: {"resolution": 1, "method":
      "cplearn",
             "snapshot_id":
                 "s_pre_1772144167_48165"}

localhost_exec:
  INFO:httpx:HTTP Request: POST
      http://127.0.0.1:41888/api/v1/
  clustering/create

function_call:
  run_clustering({
  "snapshot_id":"s_pre_1772144167_48165",
  "dataset_id":"ds_1772144073_88902",
  "project_id":"p_1772075545_60053",
  "resolution":1,
  "method":"cplearn"})

generated_snapshot_id:
     "s_cluster_cplearn_1772158045_63107"
```

Figure 6: Planning emits ID-only tool arguments; execution runs locally via localhost.

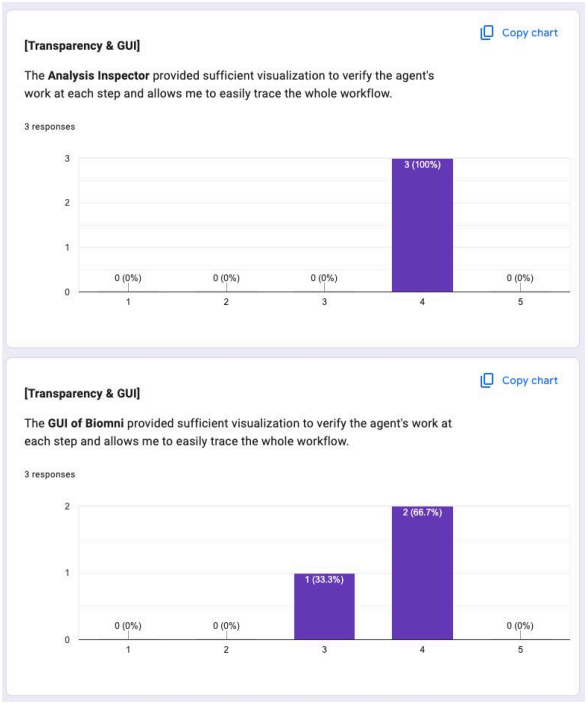
```
3. Detailed Execution Logs:
[UNKNOWN]: ...
[UNKNOWN]: ...
[UNKNOWN]: ...

4. Tool Inputs Used:
{"snapshot_id":"s_pre_1772144167_48165",
 "dataset_id":"ds_1772144073_88902",
 "project_id":"p_1772075545_60053",
 "resolution":1,
 "method":"cplearn"}

5. Generated Snapshot IDs:
"s_cluster_cplearn_1772158045_63107"

Reflection Result:
"Analysis completed successfully."
```

Figure 6: Reflection consumes ID-level tool inputs and generated snapshot identifiers while detailed execution logs are not accessible for cloud-host LLMs.

In Gardener, cloud-side planning is tethered to on-device execution via the Resident State Protocol (RSP). Under RSP, the planner operates only

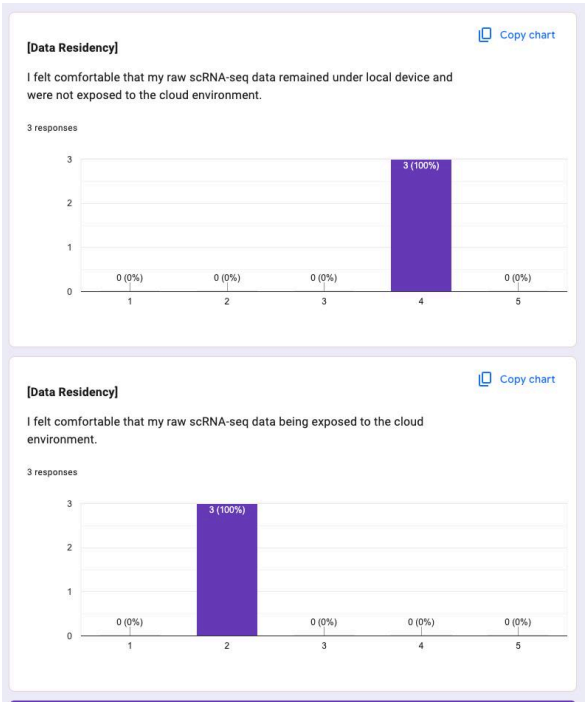
on contextual coordinates and minimal parameters, and issues tool-invocation requests that are executed locally by the Local Scientific Engine. Figure 6 shows a real execution trace: a clustering request is specified using snapshot identifiers and hyperparameters, the computation is performed via a localhost endpoint, and the resulting state transition is committed as a new snapshot. To preserve data residency, the reflection stage is also RSP-compliant: it consumes only contextual-coordinate tool inputs and generated snapshot identifiers, while detailed execution logs are withheld (reported as `[UNKNOWN]`) to avoid exposing raw artifacts, file-system pointers, or other sensitive local context.

E Questionnaire



(a) Survey regarding GUI and transparency

(b) Survey regarding steerability.



(c) Survey regarding Data Residency

Figure 7: Survey

To evaluate the practical utility and trustworthiness of Gardener, we conducted a user study in which participants interacted with the system and provided feedback through a structured questionnaire, as shown in Fig 7. The survey employed a 5-point Likert scale, ranging from 1 (Strongly Disagree) to 5 (Strongly Agree), to quantify user perceptions across three key dimensions.