BEYOND RETRIEVAL AND FINE-TUNING: LEARNING FROM SCIENTIFIC EXECUTION EXPERIENCE FOR SELF-IMPROVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific execution experience often intertwines transferable strategies with context-specific constraints. Neither imitating successful trajectories nor accumulating external experience alone guarantees effective transfer across scientific tasks. We introduce BIOMASTER, an experience-learning framework that transforms scientific execution experience into both model capabilities and reusable methodological knowledge. BIOMASTER post-trains its model on filtered execution trajectories and distills reusable scientific methods into structured Research Units (RUs), which encode applicability conditions, execution dependencies, and verification evidence to support retrieval and adaptation at inference time. A failure-guided feedback loop further constructs targeted tasks and updates future learning assets. With the base model held fixed, BIOMASTER improves over the strongest native baseline on STELLA Simple Set and Biomni-Eval1, remains competitive on BioMystery, and its separately reported Evo configuration obtains the highest BioMystery score. With the agent harness held fixed, its post-trained model improves BioMystery accuracy by 9.26 percentage points over the base model. Component analyses demonstrate that parameter learning and experience reuse are complementary, with learned capabilities increasing the utility of external experience. Together, these findings establish scientific execution experience as a valuable learning signal for building self-improving research agents.

1 INTRODUCTION

Scientific agents connect language models to specialized tools, data resources, and execution environments, as exemplified by Biomni and STELLA (Huang et al., 2025; Jin et al., 2025), and support experimental and analytical workflows in autonomous chemical research and case-based data science (Boiko et al., 2023; Guo et al., 2024). Beyond completing individual tasks, these executions generate experience for future problem solving. Existing approaches extract rules, skills, and workflows for inference-time reuse (Zhao et al., 2024; Wang et al., 2024; 2025), or train models to acquire tool-use and execution behaviors (Schick et al., 2023; Song et al., 2026). A common question remains: how can experience from one task reliably improve performance on another?

A successful scientific execution records what worked in a particular setting, but not necessarily what should transfer beyond it. Reusable strategies are intertwined with specific data, analytical assumptions, and environmental constraints. Imitating a trajectory may therefore reproduce context-dependent choices rather than generalize effective behavior. Retrieving a method does not resolve this problem either: the agent must still assess its assumptions, adapt its execution, and verify the results. Effective transfer thus requires learning reusable problem-solving behavior while retaining the conditions and evidence needed to assess when and how a method applies. Complementary Learning Systems (CLS) theory offers a useful organizing principle: rapid acquisition preserves the specifics of individual experiences, whereas gradual learning integrates shared structure (Kumaran et al., 2016). Inspired by this distinction, we consider two complementary representations of the same scientific execution experience. Trajectory-based post-training aims to internalize transferable problem-solving behavior, while structured external experience keeps methods, applicability conditions, execution dependencies, and verification evidence explicit for inspection and adaptation. Their contributions need not be independent: the value of external experience depends not only on what is retained,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

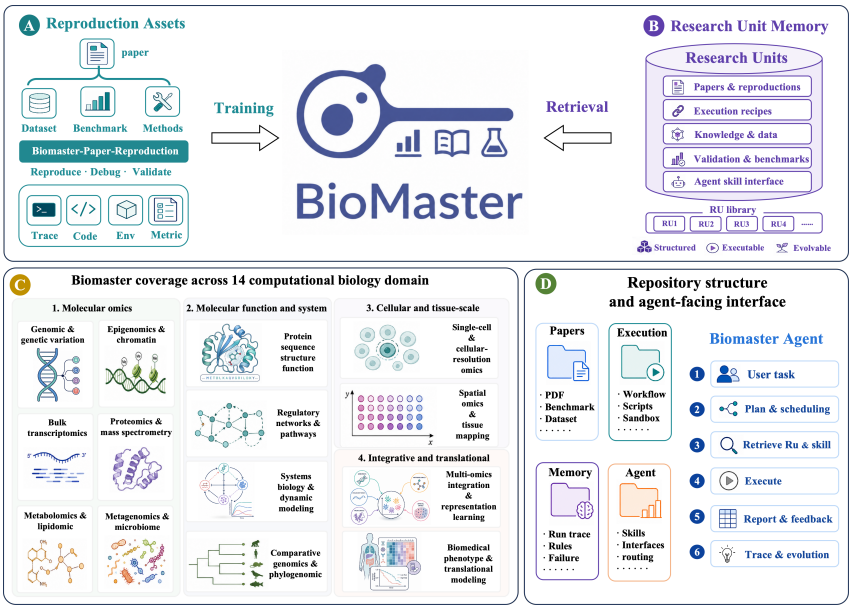


Figure 1: **Overview of BIOMASTER: complementary learning from scientific execution. (A)** Paper reproduction yields validated execution assets and trajectories for model post-training. **(B)** RUs organize methods from the same executions for retrieval and adaptation at inference time. **(C)** Coverage spans 14 computational biology domains. **(D)** An agent-facing repository supports planning, RU and skill retrieval, execution, reporting, and feedback-driven experience updates.

but also on the model's ability to interpret and apply it. This motivates our central hypothesis: post-training on scientific experience can improve not only standalone task performance, but also the benefit an agent derives from external experience.

We introduce BIOMASTER (Figure 1), a framework that couples model post-training with explicit experience reuse through a shared pool of scientific executions. Tasks constructed using a Large Knowledge Model (LKM) (Huang et al., 2026) are run within a unified execution framework, yielding trajectories that capture agent actions, tool feedback, and error correction. Task scoring and quality filtering select trajectories for post-training, while reusable methods from the same executions are distilled into Research Units (RUs). Each RU provides a self-contained method description with explicit input requirements, applicability conditions, execution dependencies, and verification evidence, supporting retrieval and adaptation at inference time. Failure diagnosis by task type further guides targeted task construction, directing subsequent experience collection toward the model's capability gaps.

Evaluations on STELLA Simple Set, Biomni-Eval1, and BioMystery support the effectiveness of this framework. Component analyses further suggest that post-training can improve not only task performance but also the benefit derived from external experience. Together, these findings highlight a central implication: the value of accumulated experience depends on the capabilities of the model using it. This motivates a direction for scientific-agent development in which experience accumulation and the ability to reuse that experience are improved together.

2 RELATED WORK

Scientific models and leading agent systems. Rapid progress in protein modeling illustrates the expanding capabilities of specialized scientific models (Jumper et al., 2021; Lin et al., 2023; Su et al., 2024; Li et al., 2024; Cheng et al., 2024; Shen et al., 2026). Using these capabilities in research requires planning, tool coordination, and result verification. ReAct, Toolformer, and ToolLLM develop tool-grounded reasoning and action (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024), while AgentBench, MLAgentBench, and ScienceAgentBench evaluate task execution beyond language-only responses (Liu et al., 2024; Huang et al., 2024; Chen et al., 2025). Biomni

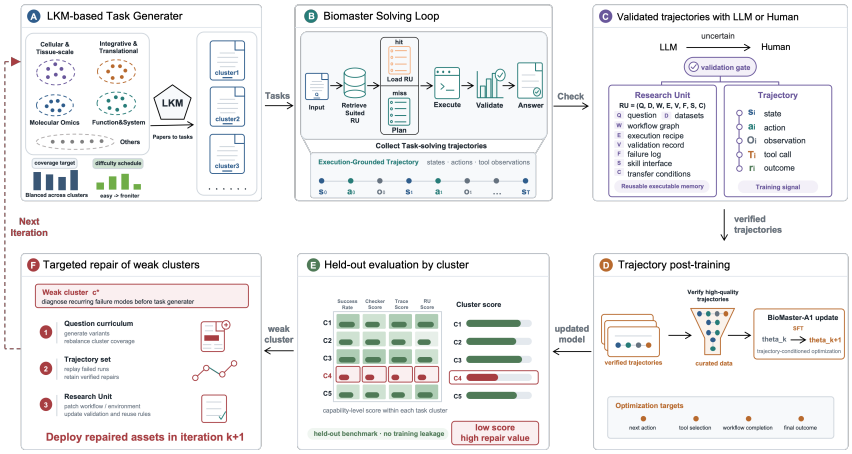


Figure 2: **BioMaster's dual-path learning loop. (A)** An LKM constructs tasks under domain-coverage and difficulty objectives. **(B)** RU-assisted execution produces tool-grounded trajectories. **(C)** Validation and distillation yield training trajectories and reusable RUs, with uncertain cases referred for human review. **(D)** Curated trajectories support model post-training. **(E–F)** Held-out evaluation identifies weak clusters and guides targeted updates to tasks, trajectories, and RUs. Solid arrows indicate forward flow; dashed arrows indicate feedback for the next iteration.

and STELLA represent leading publicly available general-purpose biomedical agents at the time of our evaluation (Huang et al., 2025; Jin et al., 2025); DS-Agent further demonstrates case-based scientific analysis (Guo et al., 2024). Our comparisons include Biomni, STELLA, and Claude Code (Anthropic, 2026b), together with Scientific Agent Skills- and BioTender-augmented Claude Code, covering strong domain-native and skill-augmented configurations. BioMaster extends this execution-centered perspective by using scientific experience to improve both model behavior and reusable methodological knowledge.

Experience reuse and complementary learning. External experience supports inference through retrieval and managed memory (Lewis et al., 2020; Packer et al., 2023), as well as reflections, rules, executable skills, and reusable workflows (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024; 2025). Post-training updates model behavior through tool-use supervision and generated trajectories (Schick et al., 2023; Song et al., 2026; Aksitov et al., 2024; Zelikman et al., 2022), or reinforcement learning and evolving curricula (Zhou et al., 2024; Carta et al., 2023; Qi et al., 2025). Related work examines self-generated feedback, reinforcement-learning dynamics, and coupled policy–memory updates (Yuan et al., 2024; Cao et al., 2026; Ye & Yu, 2026; Fu et al., 2026). BioMaster's contribution is not merely to combine training with memory, but to derive both from shared scientific executions: curated trajectories supervise model behavior, while RUs preserve local methods with applicability conditions, dependencies, and verification evidence. Crucially, controlled comparisons with a fixed RU library test whether post-training increases the benefit of experience reuse, connecting model improvement to the practical value of accumulated knowledge.

3 BioMaster

3.1 Scientific task construction

A LKM constructs literature-grounded scientific tasks under domain-coverage and difficulty objectives. Coverage spans the 14 computational biology domains in Figure 1C, grouped into molecular omics, molecular function and systems, cellular and tissue-scale analysis, and integrative and translational research. The scientific experience pool contains 10K tasks across 724 valid clusters. Tasks are deduplicated, and those overlapping external evaluation sets are excluded. Each task is packaged as a structured execution specification rather than a standalone question. The specification includes the scientific background and objective, input resources, workflow stages, execution constraints, and required deliverables. Resource manifests identify datasets and software dependencies, while

output contracts specify artifact paths, file formats, and required fields. Environment configurations define execution settings, and separate grading specifications and checkers distinguish output-format validation from task-specific result scoring. For example, a transcriptomic task requires public-data preprocessing, differential expression analysis, and pathway enrichment, with intermediate expression data and schema-defined result tables as execution artifacts. This structure connects a scientific objective to concrete execution requirements and checkable outputs.

Beyond initial collection, cluster-level failure diagnoses guide supplementary task construction, directing new experience toward capability gaps exposed by the current model.

3.2 RU-ASSISTED EXECUTION AND DUAL-ASSET EXTRACTION

The unified Harness H manages task inputs, tool interfaces, execution feedback, and RU invocation. During execution, the agent can retrieve candidate RUs, inspect their input requirements, applicability conditions, dependencies, and known failures, and adapt suitable methods. When no suitable RU is available, it develops a task-specific plan. Initial trajectories are collected using Kimi K3 as the teacher in the same Harness, with candidates sampled in groups of eight and at most 400 tool calls per trajectory.

Execution ends when the agent submits its answer and artifacts or exhausts its budget. Each run is recorded as

$$e_i = (q_i, \tau_i, A_i, v_i), \qquad \tau_i = [(a_{i,t}, o_{i,t+1})]_{t=0}^{T_i-1}, \tag{1}$$

where q_i is the task, $a_{i,t}$ an action, $o_{i,t+1}$ the resulting observation, A_i the delivered artifacts, and v_i the post-hoc acceptance and quality records.

Validation is LLM-assisted, with uncertain cases referred for human review. The collected experience $\mathcal{E}_k$ yields two complementary assets:

$$\mathcal{D}_k^{\text{sci}} = \Phi_T(\mathcal{E}_k), \qquad \widetilde{\mathcal{R}}_k = \Phi_R(\mathcal{E}_k). \tag{2}$$

Φ_T serializes trajectories for supervised learning only when both task acceptance and trajectory quality checks pass: the former evaluates answers and artifacts, while the latter assesses suitability for behavioral learning. In parallel, Φ_R extracts candidate RUs and checks their local methods against supporting code, observations, and artifacts.

Each RU captures an independently usable local method:

$$r = \langle Q, D, W, E, V, F, S, C \rangle, \tag{3}$$

encoding the local problem (Q), input requirements (D), workflow and dependencies (W), execution recipe and environment (E), verification evidence (V), known failures (F), an optional execution interface (S), and applicability conditions (C). Source tasks, trajectory fragments, and versions are retained as metadata.

Unlike final-task acceptance, RU acceptance assesses local validity and evidence coverage. A method must retain an interpretable objective, identifiable input conditions, and checkable results outside its source trajectory. A single trajectory may yield multiple RUs, while evidence from multiple executions may support the same RU.

3.3 TWO-STAGE TRAJECTORY POST-TRAINING

BioMaster-A1 is initialized from Qwen3.6-35B-A3B (Qwen Team, 2026) and post-trained in two stages, progressing from Harness adaptation to scientific problem-solving. Stage 1 applies LoRA (Hu et al., 2022) to 17,332 execution examples collected by solving simple, synthetically generated bioinformatics tasks within the BioMaster Harness. These examples provide task-grounded supervision for learning tool-call syntax, argument specification, and Harness interaction. Stage 2 continues from the adapted model with full-parameter training on 6,264 high-quality scientific trajectories, focusing on method selection, multi-step execution, result verification, and error correction. This procedure yields BioMaster-A1-v1. Both stages optimize a token-normalized masked autoregressive objective. For a serialized example $x_i = (x_{i,1}, \ldots, x_{i,L_i})$, let $\mathcal{M}_i$ denote the model-output positions selected for supervision:

$$\mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}) = -\frac{\sum_{i=1}^{|\mathcal{D}|} \sum_{t \in \mathcal{M}_i} \log \pi_\theta(x_{i,t} \mid x_{i,<t})}{\sum_{i=1}^{|\mathcal{D}|} |\mathcal{M}_i|}. \tag{4}$$

Task inputs, tool observations, and external experience used during execution are retained as conditioning context rather than supervised targets. Supervision is restricted to recorded model outputs: post-hoc acceptance records are used only for filtering, and no unrecorded reasoning is added retrospectively.

3.4 Evaluation and failure-guided repair

Held-out evaluation identifies cluster-level failures in retrieval, planning, execution, and validation. Given failure diagnoses $\mathcal{F}_k$ and current tasks $\mathcal{Q}_k$, the LKM constructs $\mathcal{Q}_{k+1} = G_{\mathrm{LKM}}(\mathcal{F}_k, \mathcal{Q}_k)$. Repair targets weak capabilities through supplementary tasks, verified recovery trajectories, and revisions to RU workflows, environments, evidence, and applicability conditions. New executions undergo the same validation and dual-asset extraction. Unlike output-level self-refinement (Madaan et al., 2023), this feedback updates future tasks and learning assets. The v1-to-v2 update targets candidate comparison, precise operations, database semantics, and result checking; evaluation assesses the combined update rather than task selection alone.

Complementarity is evaluated on 203 tasks with three executions each. Let $S(\theta, H, \mathcal{R})$ denote percentage-scale mean correctness across executions, rather than success in at least one attempt. With the Harness H fixed and a shared library $\mathcal{R}$ for RU-enabled conditions, we define

$$\Delta_{\mathrm{train}}(H) = S(\theta_{\mathrm{post}}, H, \varnothing) - S(\theta_{\mathrm{base}}, H, \varnothing), \tag{5}$$

$$\Delta_{\mathrm{RU}}(\theta) = S(\theta, H, \mathcal{R}) - S(\theta, H, \varnothing), \tag{6}$$

$$I_{\mathrm{train}\times\mathrm{RU}} = \Delta_{\mathrm{RU}}(\theta_{\mathrm{post}}) - \Delta_{\mathrm{RU}}(\theta_{\mathrm{base}}). \tag{7}$$

A positive interaction indicates greater RU benefit after post-training. Retrieval and method selection remain autonomous, so the measured RU effect encompasses retrieval, selection, adaptation, and execution.

4 Experiments

Our experiments distinguish three questions: whether BioMaster improves complete-system performance with the underlying model held fixed; whether trajectory post-training improves the model under a fixed Harness; and whether parameter learning changes the benefit of reusing external experience. These questions require different comparisons: complete-system rankings establish overall competitiveness, checkpoint comparisons characterize learned performance changes, and component contrasts test whether training and experience reuse contribute independently or interact.

4.1 Experimental setup

System-level experiments fix Claude Opus 4.6 (Anthropic, 2026c) and compare BioMaster with Claude Code (Anthropic, 2026b), Biomni, STELLA, and experience-augmented Claude Code baselines. Model-level experiments compare BioMaster-A1-v1/v2 with Qwen3.6-35B-A3B (Qwen Team, 2026) and other representative models under matched agent and scoring settings. We evaluate on STELLA Simple Set (Jin et al., 2025), Biomni-Eval1 (Huang et al., 2025), and BioMystery v11 (Anthropic, 2026a). System- and model-level evaluations use different protocols, so scores are compared only within their corresponding settings. Component experiments further isolate the effects of post-training, Harness augmentation, and RU access on a fixed set of 203 tasks. Scores are reported on a 0–100 scale, with cross-benchmark averages weighting the three benchmarks equally. Full dataset statistics, execution protocols, model and software versions, overlap checks, scoring procedures, and additional implementation details are provided in the Appendix.

4.2 System-level Harness and experience augmentation

Complete-system gains under a common backbone. Figure 3(a) evaluates whether BioMaster's complete execution and experience-reuse design remains competitive when backbone selection is controlled. Standard BioMaster obtains 97.73 on STELLA Simple Set and 83.58 on Biomni-Eval1, exceeding the strongest native-framework baseline on each benchmark by 1.95 and 3.60 points, respectively. On BioMystery, it scores 66.30, compared with 67.04 for Biomni. Its equal-weight average is nevertheless the highest among the standard configurations: 82.54, versus 80.90 for Biomni,

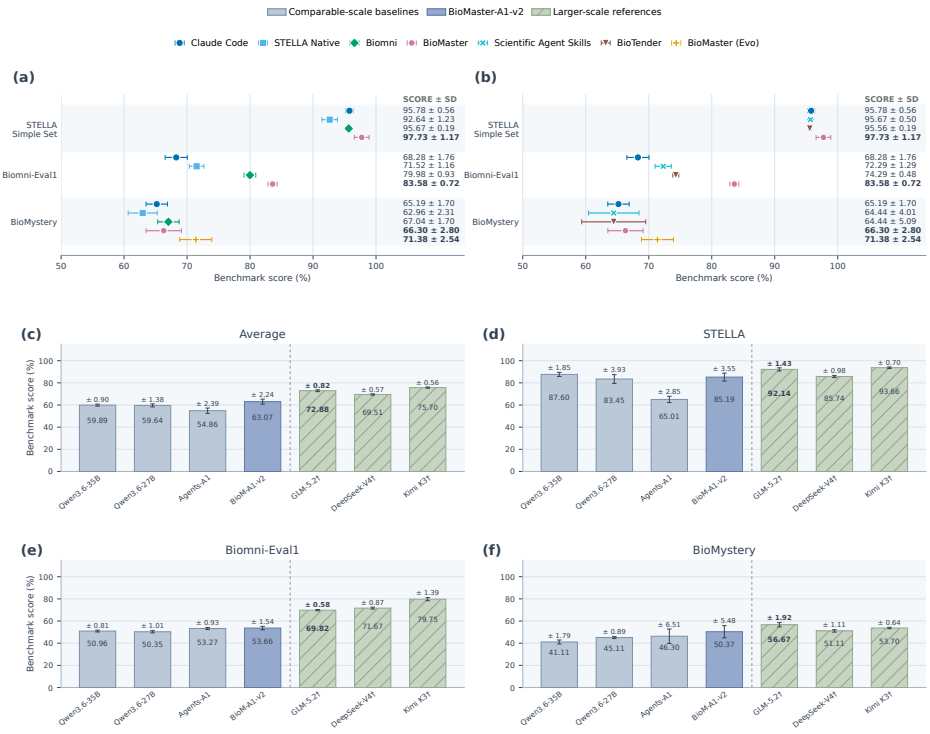


Figure 3: **System-level and fixed-Harness model comparisons. (a,b)** System-level comparisons with Claude Opus 4.6 fixed, covering complete agent frameworks and experience-augmented baselines. **(c–f)** Fixed-Harness model comparisons on the equal-weight average, STELLA, Biomni-Eval1, and BioMystery. Points and bars show means; error bars denote standard deviations. BioM-A1-v2 denotes BioMaster-A1-v2; BioMaster (Evo) is reported only in the system-level BioMystery comparison.

76.42 for Claude Code, and 75.71 for STELLA Native. The aggregate advantage therefore comes from stronger performance on two benchmarks, rather than dominance on every benchmark. The separately reported BioMaster (Evo) configuration reaches 71.38 on BioMystery, 5.08 points above standard BioMaster and 4.34 points above Biomni; this configuration-specific improvement is kept separate from the standard-system average.

Experience augmentation is not uniformly beneficial. Figure 3(b) compares BioMaster with Claude Code augmented by external skill libraries. Scientific Agent Skills and BioTender obtain averages of 77.47 and 78.10, respectively, compared with 82.54 for standard BioMaster. Relative to the stronger skill-augmented baseline on each benchmark, BioMaster gains 2.06 points on STELLA Simple Set, 9.29 on Biomni-Eval1, and 1.86 on BioMystery. The underlying pattern is informative: adding these libraries to Claude Code improves Biomni-Eval1 from 68.28 to 72.29 or 74.29, but does not improve its STELLA or BioMystery score. Thus, access to external procedural resources alone does not yield a uniform improvement across tasks. BioMaster's advantage over these configurations is consistent with the value of integrating experience reuse with execution support, but this complete-system comparison does not isolate the contribution of RUs from the rest of the Harness. The component experiments below make that distinction explicitly.

4.3 Trajectory post-training and failure-guided updating

Average gains arise from a selective change in the performance profile. Under the fixed Harness in Figure 3(c–f), BioMaster-A1-v2 achieves an equal-weight average of 63.07, exceeding Qwen3.6-35B (59.89), Qwen3.6-27B (59.64), and Agents-A1 (54.86) by 3.18, 3.43, and 8.21 points,

respectively. Against its direct base model, v2 improves Biomni-Eval1 from 50.96 to 53.66 and BioMystery from 41.11 to 50.37, while STELLA decreases from 87.60 to 85.19. The 9.26-point BioMystery improvement contributes approximately 3.09 points to the 3.18-point cross-benchmark average gain; the changes on the other two benchmarks nearly offset each other. This decomposition identifies BioMystery as the main source of the aggregate improvement, rather than suggesting a uniform increase in performance across evaluation settings. The larger-scale references retain higher averages, although the BioMystery gap to Kimi K3 is 3.33 points. These comparisons establish gains over the direct base and comparable-scale baselines without treating the larger-scale models as capacity-matched controls.

The update broadens gains rather than merely amplifying the strongest benchmark. The separately reported checkpoint-progression batch in Figure 4(a) reveals how this performance profile develops. Initial trajectory post-training raises BioMystery from 41.11 to 49.63, but lowers STELLA by 4.07 points and Biomni-Eval1 by 4.16 points. Consequently, the cross-benchmark average changes only from 59.99 to 60.09: the substantial BioMystery gain is almost offset by the other two benchmarks. The v1-to-v2 update has a different pattern. It improves all three scores, with gains of 1.62 points on STELLA, 5.31 on Biomni-Eval1, and 0.74 on BioMystery, raising the average to 62.65. Biomni-Eval1 contributes 1.77 points of this 2.56-point average improvement, whereas BioMystery contributes approximately 0.25 points. The update therefore primarily recovers performance outside the benchmark that benefited most from the initial training, while preserving and extending that initial gain. At v2, Biomni-Eval1 exceeds the base score by 1.15 points, BioMystery by 9.26 points, and the average by 2.66 points; STELLA remains 2.45 points below the base. The progression supports a more balanced final performance profile, rather than a claim that every training stage improves every benchmark.

4.4 Difficulty-stratified gains on BioMystery

Figure 4(b) locates the source of the BioMystery improvement within its two difficulty strata. From the base model to v2, Human-solvable accuracy increases from 46.58 to 59.36 (+12.78 points), while Human-hard accuracy changes from 17.65 to 11.76 (-5.89 points). Because the two strata contain 73 and 17 tasks, respectively, their contributions to the overall change are unequal: the Human-solvable improvement contributes approximately 10.37 points to overall accuracy, partly offset by an approximately 1.11-point decrease from Human-hard tasks. The reported overall gain of 9.26 points is therefore driven by improvement within the Human-solvable stratum, not by a comparable advance across the full difficulty range.

The v1-to-v2 update follows the same direction: Human-solvable accuracy rises by 1.37 points, Human-hard accuracy decreases by 1.97 points, and overall accuracy rises by 0.74 points. Together with the cross-benchmark progression, these results distinguish broader benchmark coverage from broader difficulty coverage. The update improves all three benchmark means relative to v1, but does not improve both BioMystery strata. The supported conclusion is stronger performance on tasks classified as Human-solvable, rather than evidence that the most difficult task regime has improved in parallel.

4.5 Complementarity of post-training and experience reuse

Training and execution support yield similar gains across RU-disabled settings. Figure 4(c) first tests whether post-training gains persist without external experience retrieval. Post-training improves accuracy by 5.25 points in the standard environment (50.08$\pm$2.05 to 55.34$\pm$1.50) and by 5.42 points under the BioMaster Harness (54.02$\pm$1.24 to 59.44$\pm$1.50). Conversely, the Harness adds 3.94 points for the base model and 4.11 points for the post-trained model. The reported training gains differ by only 0.17 points across environments, a descriptively near-additive pattern in this RU-disabled contrast. Thus, the training gain is not confined to the BioMaster Harness, and the Harness benefits both model states. This establishes the reference against which to interpret the much larger change in RU utility.

The effect of a fixed RU library reverses after post-training. With the Harness fixed, enabling RU retrieval reduces base-model accuracy from 54.02 to 52.71 (-1.31 points), but increases post-trained accuracy from 59.44 to 63.71 (+4.27 points; Figure 4(e)). Let $\Delta_{\mathrm{RU}}^{\mathrm{base}}$ and $\Delta_{\mathrm{RU}}^{\mathrm{post}}$ denote these RU-on

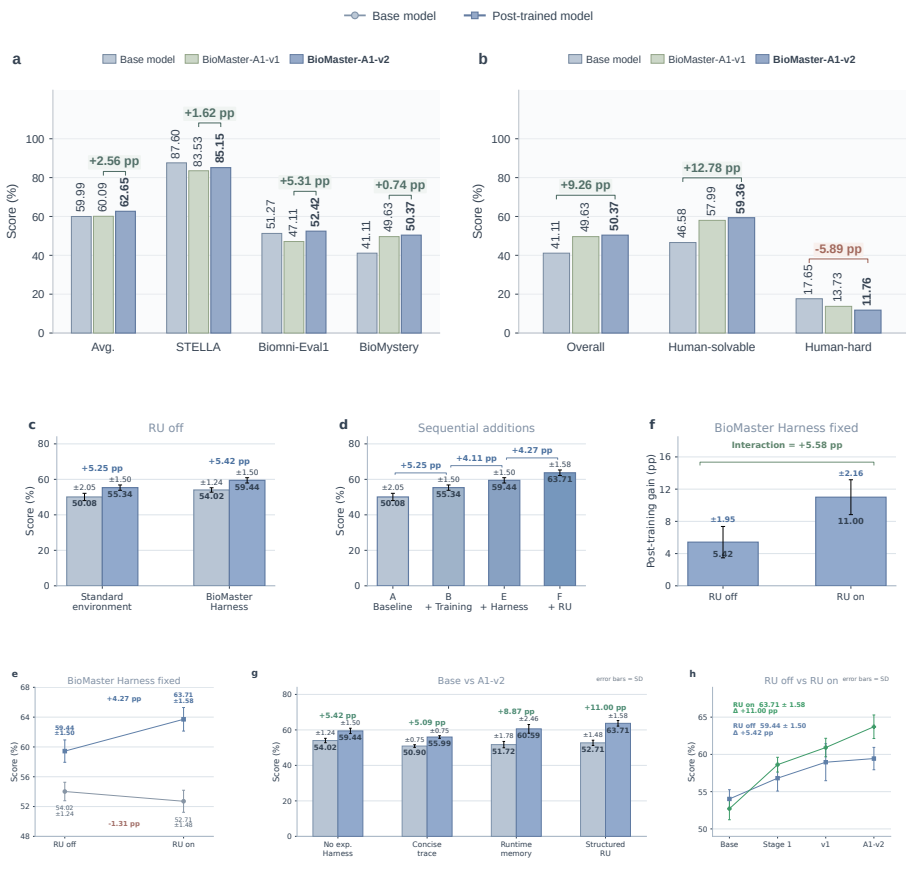


Figure 4: **Checkpoint progression and component effects. (a,b)** Performance across BioMaster checkpoints on the three benchmarks and BioMystery difficulty subsets. **(c,d)** Gains from post-training and sequential component additions. **(e,f)** RU effects under a fixed Harness and the resulting training–RU interaction. **(g,h)** Performance across experience configurations and model checkpoints. Error bars denote reported standard deviations where available; the interaction is descriptive rather than a significance test.

minus RU-off differences. Equivalently, the post-training gain increases from 5.42 points without RUs to 11.00 points with RUs (panel f). Under an additive model that retained the base-model RU effect, the post-trained RU-on score would be 58.13, rather than the observed 63.71. The difference is therefore not merely a higher score for a system with more components: training changes the sign and magnitude of the marginal benefit of experience reuse. Panel (d) summarizes one deployment path from 50.08 to 63.71 (+13.63 points), but its stepwise increments are conditional on the order of addition, not interchangeable estimates of independent component contributions.

The benefit depends on the tested experience configuration. Panel (g) shows that post-training does not make every form of external experience beneficial. Relative to the no-experience Harness, concise traces, runtime memory, and structured RUs change base-model accuracy by -3.12, -2.30, and -1.31 points, respectively. For the post-trained model, the corresponding changes are -3.45, +1.15, and +4.27 points. The important comparison is therefore not just that the trained model outperforms the base model in every configuration, but that only runtime memory and structured RUs surpass the trained model's own no-experience reference. Structured RUs attain the highest post-trained score, exceeding runtime memory by 3.12 points and concise traces by 7.72 points. Concise traces, in contrast, remain below the no-experience configuration in both model states. At the level of the tested configurations, these results support selective complementarity rather than a general benefit from providing additional experience.

Table 1: **Case study of disease-gene prioritization.**

Stage	Claude Opus 4.6 + Claude Code	Biomni	BioMaster
Task input	18 HPO phenotypes + 12 candidate genes	Same input	Same input
Initial planning	Inspects the workspace and proceeds directly to phenotype–gene reasoning	Plans HPO parsing, gene-ID mapping, and disease-database retrieval	Retrieves a reusable RU first and inspects the associated analysis procedure
Tools / resources	`ls`, `find`, followed by a research sub-agent	Reads `hp.obo`, `gene_info.parquet`, OMIM, DisGeNET, and queries Monarch	`ru.search(...)` → `ru.inspect(...)` → `ru.run(...)`
Intermediate analysis	Focuses on dental abnormalities, intellectual disability, and epilepsy	Computes phenotype overlap across candidates; e.g., OFD1 matches approximately 7/18 phenotypes and ATRX approximately 4/18	HPO normalization → candidate-gene mapping → phenotype-evidence matrix
Decision rule	Forms a disease-level hypothesis and selects an EIF2AK3-associated disorder	Primarily favors the candidate with the largest phenotype overlap, selecting OFD1	Ranks only genes within the provided candidate set using supporting and contradictory evidence
Key risk	Does not explicitly verify consistency among the inferred disease, candidate gene, and final identifier	Extensive tool use does not prevent phenotype-overlap count from becoming the dominant decision heuristic	The retrieved RU explicitly checks candidate-set constraints, contradictory phenotypes, and previously observed failure modes
Final verification	Does not detect the inconsistency between the candidate set and final prediction	Does not sufficiently verify the candidate-set constraint or required output representation	Checks candidate membership, evidence coverage, identifier mapping, and output format
Final answer	`ENSG00000116747`; score: 0	`ENSG00000046651`; score: 0	`ENSG00000060237`; correct

The RU benefit emerges across the checkpoint progression. Panel (h) adds a temporal view of this interaction. RU-on begins below RU-off for the base model, is already above it at Stage 1, and remains higher through v1 and v2, ending with a 4.27-point advantage. Across the full progression, the RU-on score increases by 11.00 points, compared with 5.42 points for RU-off. The increased utility is therefore visible before the final update and develops while the RU library is held fixed, rather than requiring an expanded library. This is a model–retrieval-system result: fixing the library does not imply that different checkpoints retrieve or use identical evidence. Taken together, the component comparisons distinguish two relationships. Execution support yields similar gains across the two model states without RUs, whereas the benefit of experience reuse changes substantially with training and with the experience configuration. The evidence for complementary learning is thus not simply that training and RUs both improve the final system, but that training changes the conditions under which external experience becomes beneficial.

4.6 Case Study: Constraint-Aware Gene Prioritization

We further compare the execution behavior of Claude Opus 4.6 with Claude Code, Biomni, and BioMaster on a representative disease-gene prioritization task with 18 HPO phenotypes and 12 candidate genes. Claude Code forms a plausible disease-level hypothesis but does not preserve the closed candidate-set constraint through final identifier generation. Biomni performs substantially more structured retrieval, yet its decision is dominated by phenotype-overlap counts. BioMaster instead retrieves a reusable RU that explicitly structures candidate mapping, supporting and contradictory evidence, and final identifier verification. The RU therefore provides procedural guidance rather than the answer itself, illustrating how reusable external experience complements parametric reasoning during scientific execution.

5 Discussion and Conclusion

BioMaster converts verified executions into two complementary forms of reusable experience: post-training trajectories that internalize recurring reasoning patterns and Research Units (RUs) that preserve explicit procedural knowledge. The 5.58 percentage-point increase in RU reuse gain after post-training shows that these mechanisms are complementary, enabling scientific agents to improve through accumulated research experience rather than repeatedly solving tasks from scratch.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and polishing of the BioMaster manuscript, including improvements to grammar, wording, clarity, and consistency.

ETHICS STATEMENT

This study uses computational biology tasks and scientific resources and does not collect new human-subject data. The execution environment isolates model-visible inputs from verifier-only references. External datasets and software are used subject to their respective licenses and benchmark terms.

REPRODUCIBILITY STATEMENT

The main text describes the model comparisons, benchmarks, and evaluation settings. Appendix A documents task construction, data splits, and the RU schema; Appendix B documents Harness isolation, execution budgets, and a worked RU example; Appendix C documents training data, hyperparameters, loss masking, and feedback-driven updates; and Appendix D documents evaluation protocols, answer parsing, scoring, and software configuration.

REFERENCES

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix Yu, and Sanjiv Kumar. ReST meets ReAct: Self-improvement for multi-step reasoning LLM agent. In *International Conference on Learning Representations*, 2024.

Anthropic. Evaluating claude's bioinformatics research capabilities with BioMysteryBench. Anthropic Research, April 2026a.

Anthropic. Claude code by anthropic. Anthropic product page, 2026b.

Anthropic. Introducing claude opus 4.6. Anthropic News, 2026c.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 2023. doi: 10.1038/s41586-023-06792-0.

Hanqun Cao, Hongrui Zhang, Junde Xu, Zhou Zhang, Lingdong Shen, Minghao Sun, Ge Liu, Jinbo Xu, Wu-Jun Li, Jinren Ni, Cesar de la Fuente-Nunez, Tianfan Fu, Shuting Jin, Pheng-Ann Heng, and Fang Wu. The forgetting-learning trade-off: Making reinforcement learning work for protein language models. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 10520–10531, 2026.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025.

Xingyi Cheng, Bo Chen, Pan Li, Jing Gong, Jie Tang, and Le Song. Training compute-optimal protein language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zenghuang Fu, Zhaoyang Li, Qiuyuan Ai, Haoyu Wu, Minghui Wu, Chenxu Zhao, Ante Wang, Guannan He, and Changwei Wang. Self-play meets skill evolution: Self-evolving search agents that pose, solve, and remember. *arXiv preprint arXiv:2607.29468*, 2026.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated data science by empowering large language models with case-based reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16813–16848. PMLR, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael Snyder, Le Cong, Aviv Regev, and Jure Leskovec. Biomni: A general-purpose biomedical AI agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024.

Yuan Huang, Sihan Hu, Hongyu Gu, Chao Ma, Jiaxing Zhang, Zhiyong Zou, Caiyu Fan, Yan Xiao, Mingjun Xu, Chenyu Xie, Mingzhen Ju, Zhehao Ma, Qi Zhang, Baozong Wang, Yu Li, Zhiyuan Yao, Ruoxue Liao, Xinyu Li, Linfeng Zhang, Kun Chen, and Weinan E. Large knowledge model: From papers to a scientific reasoning landscape. *arXiv preprint arXiv:2609.27297*, 2026. doi: 10.48550/arXiv.2609.27297.

Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. STELLA: Self-evolving LLM agent for biomedical research. *arXiv preprint arXiv:2507.02004*, 2025.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, Ellen Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michal Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021.

Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 2016. doi: 10.1016/j.tics.2016.05.004.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Mingchen Li, Yang Tan, Xinzhu Ma, Bozitao Zhong, Huiqun Yu, Ziyi Zhou, Wanli Ouyang, Bingxin Zhou, Pan Tan, and Liang Hong. Prosst: Protein language modeling with quantized structure and disentangled attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan Dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all. Qwen model release, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Lingdong Shen, Linlin Chao, Tang Liu, Qi Liu, Guowei Zhou, Hao Wang, Xinyue Dong, Tianhong Li, Xiaoming Zhang, and Jinren Ni. Proteinsage: From implicit learning to explicit structural constraints for efficient protein language modeling. *bioRxiv*, 2026. doi: 10.64898/2026.03.17.712034.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Yueqi Song, Ketan Ramaneti, Zaid Sheikh, Ziru Chen, Boyu Gou, Tianbao Xie, Yiheng Xu, Danyang Zhang, Apurva Gandhi, Fan Yang, Joseph Liu, Tianyue Ou, Zhihao Yuan, Frank F. Xu, Shuyan Zhou, Xingyao Wang, Xiang Yue, Tao Yu, Huan Sun, Yu Su, and Graham Neubig. Agent data protocol: Unifying datasets for diverse, effective fine-tuning of LLM agents. In *International Conference on Learning Representations*, 2026.

Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. In *International Conference on Learning Representations*, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 63897–63911, 2025.

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

Shicheng Ye and Chao Yu. Joint learning of experiential rules and policies for large language model agents. *arXiv preprint arXiv:2606.27136*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Table 2: Search and organization scale of the LKM pipeline.

Stage or object	Count	Meaning
Initial domain seeds	14	Predefined domain descriptions
Expansion/frontier seeds	192	Semantic retrieval expansion terms
Literature reasoning chains	200,758	Retrieved literature-level chains
Paper–workflow records screened	254,883	Records assessed for relevance
Relevant records retained	216,534	Records supporting at least one workflow family
P0 strong candidates	117,017	Records passed to full-text extraction
Workflow families	1,770	Deduplicated family-level structures
Paper–workflow records in final candidate set	116,928	Family-linked records after organization
Deduplicated papers	99,951	Unique paper identifiers

A Data and task sources

The statistics below use the September 2026 dataset snapshot. Papers, workflow instances, and unified workflow clusters are treated as distinct units throughout.

A.1 LKM-driven task construction

We construct literature-grounded scientific tasks with a Large Knowledge Model (LKM) pipeline. A workflow family is the intermediate organization unit: generic workflows and their technical variants provide the structural template, article-level workflows and candidate tasks provide the scientific setting, and domain coverage and research activity determine which sources receive priority. The pipeline has four stages: domain retrieval, paper–workflow screening, targeted workflow selection, and paper-level task instantiation under workflow constraints. Figure 5 summarizes the counts at the retrieval and organization stages.

Domain retrieval and candidate workflows. The initial retrieval covers genomic variation, bulk and single-cell transcriptomics, ChIP-seq and chromatin analysis, proteomics and mass spectrometry, protein structure prediction, microbiome analysis, metabolomics, multi-omics integration, biomedical machine learning, survival analysis, and systems and network biology. Fourteen domain descriptions provide the initial seeds, and 192 expansion or frontier seeds are used during semantic retrieval. This stage returns 200,758 literature reasoning chains.

Paper–workflow screening. Screening is performed on paper–workflow correspondences rather than on paper IDs alone. We evaluate 254,883 correspondences, retain 216,534 relevant records, and mark 117,017 as P0 strong candidates for full-text extraction. The resulting candidate inventory contains 1,770 workflow families, 116,928 paper–workflow records, and 99,951 papers after deduplication by paper ID. P0 denotes a strong candidate for review, P1 denotes relevant but incomplete evidence, and P2 denotes a weak or boundary case. These labels organize review and do not measure paper quality or task difficulty.

Activity and coverage. Workflow priority combines a recent-window weighted score, the number of records in the recent window, the 2026 cross-section count, and a trend ratio. The recent window spans 2022–2026; the 2026 value covers a partial year. The score is used to balance persistent activity, current volume, and growth. For example, Cox–LASSO prognostic feature construction (WF-4) has 394 recent-window records and remains a priority despite no 2026 record; single-nucleus RNA-seq atlas construction and cell-type annotation (WF-110) has 162 recent-window and 13 cross-section records; and multimodal biomedical representation and fusion (WF-1137) has 83 recent-window and 11 cross-section records. No single annual count determines selection.

Coverage is aligned to 14 predefined domains and their sub-directions using reproducible keyword and rule matches. The domain taxonomy used in the inventory is: (i) genomic and genetic variation, (ii) epigenomics and chromatin, (iii) bulk transcriptomics, (iv) proteomics and mass spectrometry, (v)

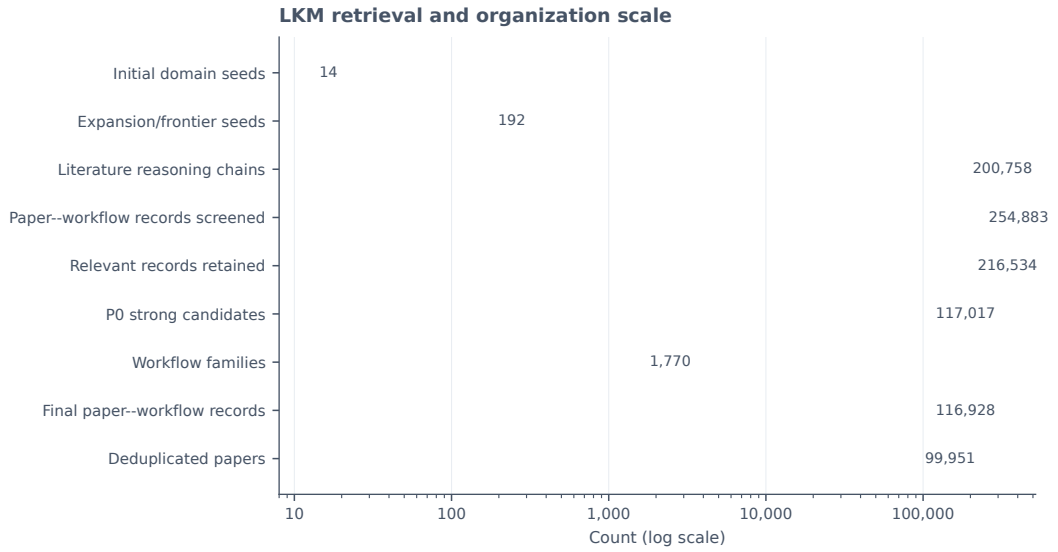


Figure 5: Absolute counts in the LKM retrieval and organization pipeline. The logarithmic horizontal axis keeps both seed counts and large record inventories readable; values are printed on the bars.

metabolomics and lipidomics, (vi) metagenomics and microbiome analysis, (vii) protein sequence, structure, and function, (viii) regulatory networks and pathways, (ix) systems biology and dynamic modeling, (x) comparative genomics and phylogenomics, (xi) single-cell and cellular-resolution omics, (xii) spatial omics and tissue mapping, (xiii) multi-omics integration and representation learning, and (xiv) biomedical phenotype and translational modeling.

Workflow-constrained instantiation. A task is instantiated from the correspondence between a workflow-level structure and a paper-level instance, rather than by expanding a domain name into a generic question. Each workflow family stores a definition, a workflow skeleton, and technical variants in `family.json`; canonical workflow templates are stored in `workflow_templates.json`. `family_index.json` records supporting paper identifiers, relative paths, and priority. Paper-level records retain the paper's workflow, its mapping to the family, and candidate tasks. A resource registry links tools, datasets, and models to the required steps.

A candidate becomes a complete task only when its paper-level workflow exposes identifiable inputs, outputs, and validation criteria. A generic family skeleton without these paper-level fields remains a candidate workflow and is not promoted to a complete task. These identifiers preserve provenance. For transcriptomics, for example, WF-110 denotes atlas construction and cell-type annotation, WF-390 compares cell-type annotation methods, WF-109 is a single-cell batch-correction benchmark, and WF-48 is bulk RNA-seq differential expression. These remain distinct task objectives even though they share a broad domain.

A.2 Trajectory collection, quality screening, and dual-asset extraction

We use Kimi K3 in the BioMaster Harness to execute each scientific task. Eight candidate trajectories are sampled per task and a single trajectory may use at most 400 tool calls. The model chooses tools, sets parameters, runs code, inspects intermediate results, and recovers from execution errors using the current environment feedback. A run ends when the requested answer and artifacts are submitted or when its execution budget is exhausted.

For task q_i and candidate run j, we record

$$e_{i,j} = (q_i, \tau_{i,j}, A_{i,j}, v_{i,j}) \,, \tag{8}$$

where $\tau_{i,j}$ is the time-ordered model/tool interaction, $A_{i,j}$ is the submitted artifact set, and $v_{i,j}$ is the post-execution acceptance and quality record. Eight is a candidate sampling size, not a claim that eight runs succeed or enter training.

Table 3: Quality gate applied to candidate execution records.

Check	Object	Admission rule
Score validity	Individual run and checker record	Reject anomalous or uninterpretable scores; a high score alone is not sufficient.
Group stability	Eight-run candidate group	Reject groups whose score variation exceeds the active stability criterion.
Score discrimination	Eight-run candidate group	Reject groups whose scores do not separate candidate quality sufficiently for selection.
Completion	Answer and required artifacts	Reject runs that stop before the task's final answer or mandatory deliverables are produced.

Table 4: Mapping from an accepted execution to a training example.

Recorded component	Training treatment	Rationale
System and user messages	Context; loss mask $= 0$	Preserve task requirements without supervising the prompt.
Tool observations	Context; loss mask $= 0$	Condition later actions on actual environment feedback.
Assistant reasoning	Target only when recorded	Preserve observed reasoning without inventing hidden steps.
Assistant tool calls	Target only when recorded	Learn the recorded calls and their arguments.
Assistant final content	Target when recorded	Learn the submitted answer and delivery description.
Quality and checker fields	Post-hoc filter only	Do not leak acceptance information into the solving sequence.

The quality gate combines individual-run acceptance with group-level evidence. A group-level check does not automatically admit every member of the group; an individual trajectory must also have an acceptable answer, artifact set, and behavior record. Conversely, an execution can contain failed attempts and still be admitted if the recorded interaction shows recovery and the final artifacts pass the task-specific checks.

A.2.1 Conversion to the Qwen training format

Accepted trajectories are serialized as multi-turn interactions. The original order of model actions, tool returns, and later operations is preserved rather than compressing the run into a reference answer. System and user messages, tool observations, and externally retrieved experience are conditioning context. Supervision is applied only to model outputs that are present in the record: actual reasoning content, actual tool calls, and the actual final answer. A missing reasoning field remains missing; checker or human review never creates retrospective reasoning.

A.2.2 Extraction of structured Research Units

In parallel with training serialization, the same accepted executions are distilled into independently usable Research Units (RUs). An RU is a local method with a clear problem boundary, input and output contract, executable conditions, and validation evidence. Source task IDs, trajectory spans, and versions remain as provenance metadata, but the method does not require the source conversation, source temporary directory, or a BioMaster-specific tool name in order to be understood.

$$r = \langle Q, D, W, E, V, F, S, C \rangle, \tag{9}$$

where Q is the local question, D the data and preprocessing requirements, W the workflow and dependencies, E the executable recipe and environment, V the validation evidence, F the observed failures and recovery actions, S the optional skill interface, and C the transfer conditions.

Table 5: Research Unit extraction fields.

Field	Name	Minimum retained content
Q	Local question	Target operation, output, domain, and modality.
D	Data	Identifiers, versions, provenance, preprocessing, and input schema.
W	Workflow	Dependencies, branches, decision points, and termination criteria.
E	Execution	Tools, versions, commands, parameters, environment, and resources.
V	Validation	Acceptance rule, check output, evidence link, validator, and uncertainty.
F	Failures	Observable symptom, cause when known, invalid output, and recovery action.
S	Skill interface	Optional callable name, inputs, outputs, and composition contract.
C	Transfer conditions	Applicability, invalid contexts, staleness, and safety restrictions.

Table 6: Distribution of workflow instances per source paper.

Workflows per paper	Papers	Paper share	Implied instances
1	9,136	92.60%	9,136
2	651	6.60%	1,302
3	69	0.70%	207
4	7	0.07%	28
5	2	0.02%	10
6	1	0.01%	6
Total	9,866	100.00%	10,689

A failure field describes a condition and the recorded response, not only an error string. Runtime images and parameters remain tied to the execution conditions that support them. Validation rules for a new input are stored separately from the historical evidence that supported the RU. An RU is accepted only when its local goal, inputs, outputs, and checks remain interpretable after the source trajectory is removed. One trajectory can yield multiple RUs, and evidence from multiple executions can support one RU.

A.3 Data distribution and partitioning

The inventory contains 9,866 deduplicated papers, 10,689 workflow or task instances, and 724 unified workflow clusters. These units have different roles: papers supply research context, workflow instances define executable tasks, and clusters organize related workflow types.

The implied-instance column is obtained by multiplying the number of papers in each bin by the corresponding number of workflows. The six-bin distribution is the frequency distribution for this inventory.

Modalities and workflow coverage. The inventory spans transcriptomics and gene expression, clinical and epidemiological analysis, genomic variation, proteomics, ecology and evolution, epigenomics, metabolomics, microbiome analysis, imaging, and structural modeling. The modality table is labelled "Top 5" in the inventory but contains four categories. Its percentage columns use modality-specific denominators, whereas the inventory totals used elsewhere in this appendix are 10,689 instances and 724 clusters. The percentages are therefore not recomputed against the full inventory.

Workflow length. Among instances with structured workflow steps, the mean length is 6.8 and the median is 6. The four bins below cover the structured workflow instances; the 4–9 step range therefore sums to 74.89%.

Component evaluation split. The component evaluation set is constructed by domain organization, cluster filtering, and one-task-per-cluster extraction. Let $\mathcal{Q}_c$ be the task set in cluster c and $n_c = |\mathcal{Q}_c|$.

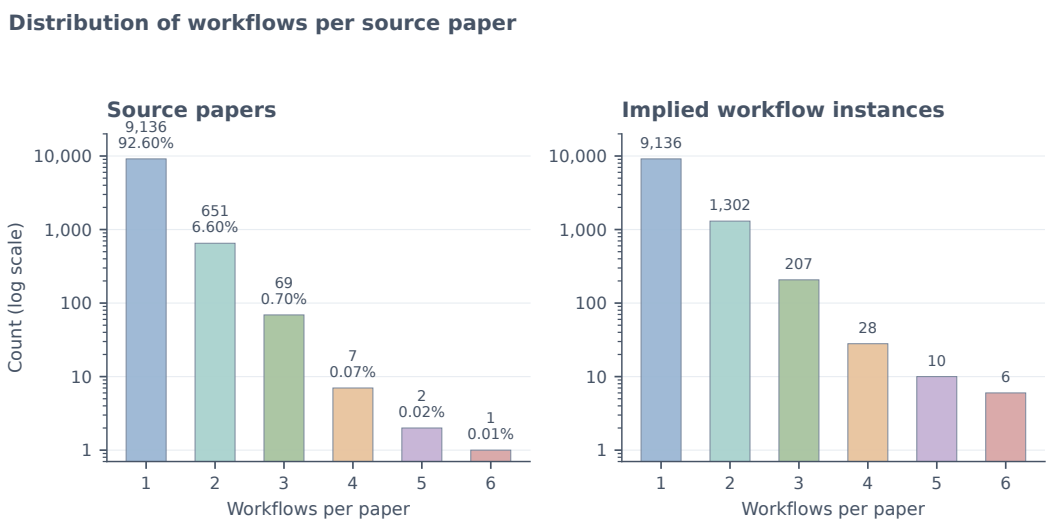


Figure 6: Paper-to-workflow relation. The two panels show source-paper counts and the corresponding implied workflow-instance counts for papers supporting one through six workflows.

Table 7: Modality ranking and workflow-cluster coverage.

Modality	Tasks	Task share	Clusters	Cluster share
Transcriptomics and gene expression	1,382	17.94%	281	28.38%
Clinical and epidemiological data	1,180	15.32%	120	12.12%
Genomics and genetic variation	1,018	13.21%	231	23.33%
Proteomics and mass spectrometry	465	6.04%	97	9.80%

Eligible clusters satisfy

$$\mathcal{C}_{\text{eligible}} = \{c : n_c > 4\}. \tag{10}$$

For the selected set $\mathcal{C}_{\text{selected}} \subseteq \mathcal{C}_{\text{eligible}}$, one task $q_c \in \mathcal{Q}_c$ is sampled per cluster:

$$\mathcal{Q}_{\text{eval}} = \{q_c : c \in \mathcal{C}_{\text{selected}}\}, \qquad |\mathcal{Q}_{\text{eval}}| = 203. \tag{11}$$

This prevents a large cluster from receiving multiple evaluation weights merely because it contains more tasks.

The individual domain counts range from 8 to 25 tasks (3.94% to 12.32%). This split describes cluster-level sampling and does not by itself assign a task to training or to RU construction.

B Harness, Research Units, and Execution Environment

B.1 Execution Environment

The fixed-model environment comparison uses the same model, Harbor task package, public inputs, task-level resource limits, and official scorer in the standard Harbor environment and in the BioMaster Harness. Both environments permit code execution, file operations, and process management. Their controlled difference is task initialization, tool orchestration, optional experience access, model-call logging, and result collection.

The BioMaster Harness runs a pinned prebuilt image containing the BioMaster runtime configuration, Python 3.12-compatible scientific dependencies, the native execution runtime, and the local RU package. The model endpoint, model identifier, and credentials are injected at runtime and are not stored in the image or task package. The standard environment uses the task image and runner configuration and does not implicitly include the local RU package.

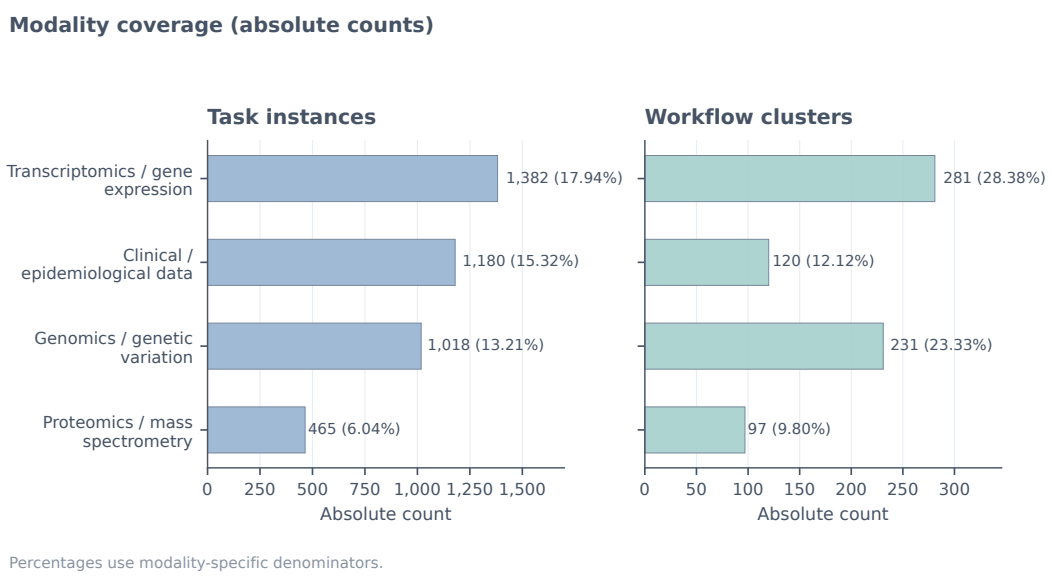


Figure 7: Task and workflow-cluster counts for the modality analysis. Labels give absolute counts and modality-specific percentages.

Table 8: Workflow-step distribution for instances with structured steps.

Step bin	Share	Note
1–3	9.68%	Short workflows.
4–6	41.26%	Middle-length workflows.
7–9	33.63%	Longer multi-step workflows.
10 or more	15.43%	Longer workflows.
Mean / median	6.8 / 6	Percentages are over structured workflow instances.

B.1.1 Task adaptation and tool interface

The standard Harbor runner mounts the task description, public data, and task working directory. The agent reads the task through `instruction.md` and the public files. Before execution, the BioMaster Harness validates the task package and generates an isolated public-input directory, task configuration package, and run profile. The public task description is kept as a file and is also copied into the first user message under the `<harbor_public_question>` field.

B.1.2 Layered budgets and permissions

Generation, agent lifetime, worker processes, and sandbox lifetime are tracked as separate budgets. A task-level CPU, memory, writable-space, or time limit that is stricter than a controller default takes precedence. The controller defaults for this environment are given in Table 11; task-level limits remain authoritative when they are more restrictive.

The limits operate at different scopes. For a shared object, the most restrictive active limit takes precedence; task-specific CPU, memory, writable space, and time limits therefore override the defaults above.

The agent runs as a non-root user. Scratch directories are isolated by run ID and the scratch root uses sticky permissions. An agent cannot elevate privileges, inspect another sandbox's processes, or take over a long-running process in another sandbox. Model requests pass through a local forwarding layer; credentials are redacted when traces are exported.

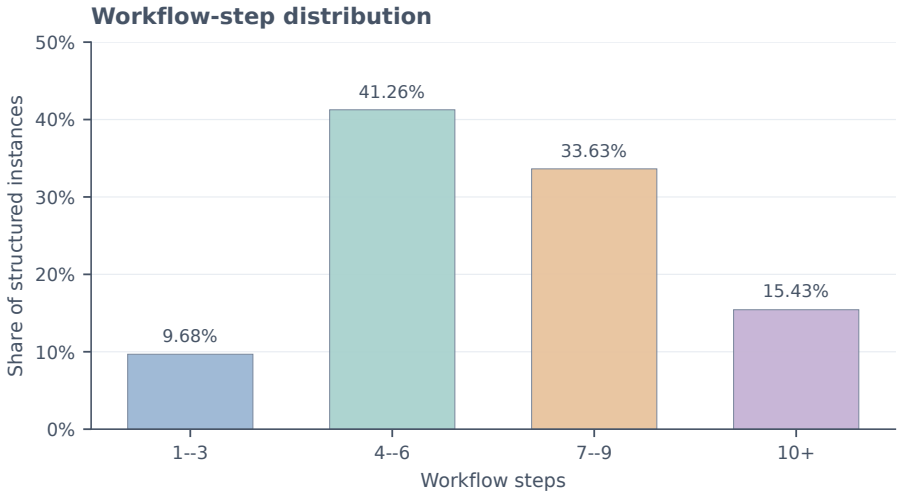


Figure 8: Workflow-step distribution for structured workflow instances.

Table 9: Domain distribution for the 203-task component evaluation set.

Domain group	Tasks	Share
Transcriptomics, RNA-seq, and differential expression	22	10.84%
Networks, pathways, enrichment, and systems biology	25	12.32%
Protein sequence, structure, and function prediction	20	9.85%
Phylogenetics, classification, evolution, and population genetics	20	9.85%
Statistical modeling, machine learning, and optimization	13	6.40%
Pathogens, microbial genomes, and antimicrobial resistance	16	7.88%
Cancer, disease, and clinical computation	13	6.40%
Variants, genetic association, GWAS, and QTL	12	5.91%
Genome assembly, annotation, and comparative genomics	11	5.42%
Plant and crop studies	12	5.91%
Epigenomics, regulation, ATAC-seq, ChIP-seq, and methylation	8	3.94%
Single-cell and spatial omics	10	4.93%
Microbiome, metagenomics, and 16S	11	5.42%
Proteomics, metabolomics, and mass spectrometry	10	4.93%
All selected tasks	203	100.00%

B.1.3 Access control, scoring isolation, and result collection

The agent receives only the public question, public inputs, and required runtime resources. Private tests, reference answers, calibration material, and hidden scoring logic are accessible only to the verifier. The public `ValidateOutputs` interface is read-only and does not return hidden answers or private score feedback.

A run follows this sequence: task-package validation, profile creation, agent execution, artifact collection and public structure checks, trace export, independent official scoring, and archival cleanup. The runtime records the model-visible system, user, assistant, and tool messages together with tool arguments, tool returns, finish reasons, usage, and exceptions. Streaming responses are reassembled before secrets and other sensitive fields are removed.

If trace or artifact download fails, the controller retains the sandbox for recovery and audit rather than deleting it immediately. Training-format export is a candidate record; admission to post-training still requires the quality checks in Appendix A.2.

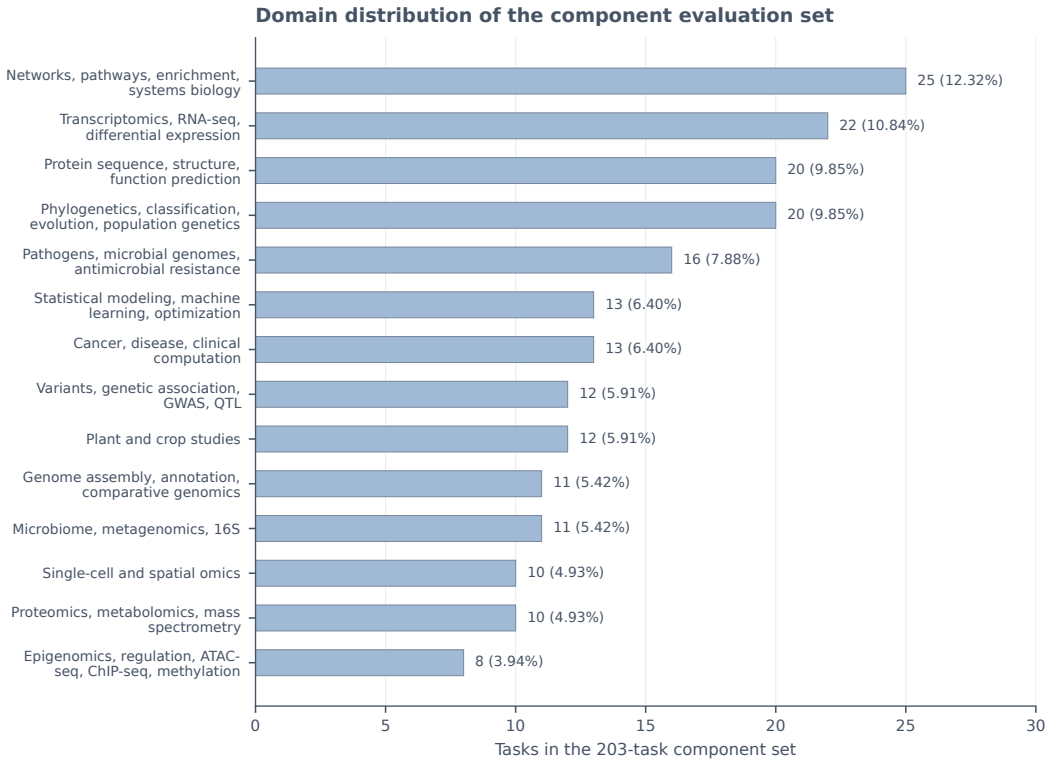


Figure 9: Absolute task counts and shares across the 14 component evaluation domains.

Table 10: Standard Harbor and BioMaster Harness configuration.

Component	Standard Harbor	BioMaster Harness
Task initialization	Runner mounts task files and public inputs	Adapter validates the package and builds an isolated public-input directory and run profile
Agent-visible question	Task files and runner-provided prompt	Same public task plus the explicit `harbor_public_question` block
Experience access	Determined by the native runner	RU access is enabled only for the configured condition
Runtime and tools	Task image and runner configuration	Pinned BioMaster image, runtime configuration, and local RU package
Model service	Runner-defined endpoint and routing	Endpoint, model ID, and credentials injected at runtime
Scoring access	Independent official scorer	Independent verifier; private tests and references remain outside the sandbox

B.2 RU CONTENT AND WORKED EXAMPLE

An RU is a self-contained scientific method. It states the local objective, input and output contract, workflow, executable environment, validation criteria, observed failure modes, and transfer conditions. The representation is decoupled from any particular agent framework: a user maps file reads, commands, and checks to the tools available in its own runtime.

Table 13 instantiates these fields for one concrete bulk RNA-seq differential-expression RU. A representative invocation supplies “counts.tsv”, “samples.tsv”, and an explicit treated-versus-control contrast. The RU filters genes, fits the DESeq2 negative-binomial generalized linear model, applies the contrast and multiple-testing correction, and writes “de_results.tsv” together with a run record.

Table 11: Default execution budgets and resource settings.

Budget level	Default setting	Interpretation
Single generation	`maxTokens = 32768`	Upper bound on tokens emitted in one generation; it is not a 32K total-context limit.
Agent runtime	`maxRuntimeSeconds = 36000`	Upper bound on the complete agent interaction.
Sandbox compute	`2c4g`	Default controller resource specification.
Worker timeout	39,600 s	Maximum duration of one worker execution.
Sandbox timeout	86,400 s	Maximum lifetime of the isolated sandbox.

Table 12: Execution outputs and retention policy.

Output	Stored content	Isolation or use
Task artifacts	Declared files and output metadata	Public structure checks run before archive.
Model-visible trace	Messages, tool calls, returns, finish reason, usage, and errors	Does not contain verifier-only references; written with restrictive permissions.
Runtime metadata	Run ID, profile, environment, and lifecycle events	Supports audit and recovery.
Official score	Verifier and scorer output	Stored separately; not fed back into the solving context.
Sandbox record	Archive and cleanup status	Sandbox is removed only after trace, outputs, and score are downloaded and checked.

The output is accepted only when the sample groups, result columns, finite statistics, and sample-count checks pass. Missing group labels, duplicated sample identifiers, or an under-replicated design trigger the recorded correction or rejection path. The method therefore has a precise transfer boundary: it applies to bulk RNA-seq count data with biological replicates, and it is not transferred to incompatible input types. The source task directory and conversation are not required for this invocation.

B.3 External experience controls

Under a fixed BioMaster Harness, the base model and BioMaster-A1-v2 are configured with no external experience, concise text traces, runtime memory, or structured RUs. All conditions use retrieval when an experience source is enabled; the comparison concerns what is stored and exposed, not whether a retrieval mechanism exists.

C Post-training and feedback updates

C.1 Two-stage data and checkpoints

BioMaster-A1 starts from Qwen3.6-35B-A3B. Stage 1 applies LoRA to 17,332 execution examples obtained from synthetic, simple bioinformatics tasks in the BioMaster Harness. The objective is to learn tool-call syntax, argument organization, and Harness interaction. Stage 2 uses full-parameter training on high-quality scientific trajectories, followed by the feedback-guided update that produces BioMaster-A1-v2. The detailed stage records list 4,264 trajectories for Stage 2 and 6,264 for Stage 3, whereas the two-stage summary uses 6,264 as the Stage-2 total. Both entries are retained in Tables 17–19; exact data accounting should be checked against the training manifest.

The two-stage summary corresponds to BioMaster-A1-v1; the detailed training tables preserve the separate Stage-2 and feedback-update entries used for the later model versions.

Table 13: Concrete Research Unit example for bulk RNA-seq differential expression. The table instantiates one RU rather than reporting an aggregate RU count.

Field	Example value	What the field makes explicit
Q	Differential expression between two groups	Scientific target and comparison.
D	Count matrix, sample sheet, gene IDs, reference annotation	Input schema and preprocessing assumptions.
W	Design formula, filtering, DESeq2 fit, contrast, multiple-testing correction	Method steps and dependency order.
E	Pinned R/DESeq2 environment; explicit input and output paths	Reproducible execution boundary.
V	Required columns, finite statistics, sample-count checks, and output schema	Validation of the new result.
F	Missing group labels, duplicated sample IDs, or low-replicate design with recorded recovery	Known failure and correction behavior.
S	Optional callable interface accepting paths and contrast	Framework-independent invocation contract.
C	Applicable to bulk RNA-seq counts with biological replicates; invalid for incompatible input types	Transfer boundary.

Table 14: External experience conditions.

Condition	Indexed material	Information available at inference
No experience	None	The task, tools, and model's parametric knowledge.
Concise text trace	Compressed trajectory text	Historical operations that the agent must interpret and adapt itself.
Runtime memory	Environment and runtime fields extracted from RUs	Images, dependencies, parameters, interfaces, and recorded runtime recovery.
Structured RU	Full self-contained RU	Scientific objective, inputs, workflow, environment, validation, failures, skill interface, and transfer conditions.

C.2 Serialization and masked supervision

Both stages optimize a token-normalized masked autoregressive objective:

$$\mathcal{L}_{\mathrm{SFT}} = -\frac{\sum_i \sum_{t \in M_i} \log \pi_\theta(x_{i,t} \mid x_{i,<t})}{\sum_i |M_i|}, \tag{12}$$

Table 15: Two-stage post-training data and checkpoint semantics.

Stage	Starting checkpoint	Examples	Update	Primary capability
1	Qwen3.6-35B-A3B	17,332	LoRA	Tool syntax, arguments, and Harness interaction.
2	Stage-1 checkpoint	6,264	Full parameters	Method choice, multi-step execution, verification, and correction.
Output	–	–	–	BioMaster-A1-v1 contains both stages.

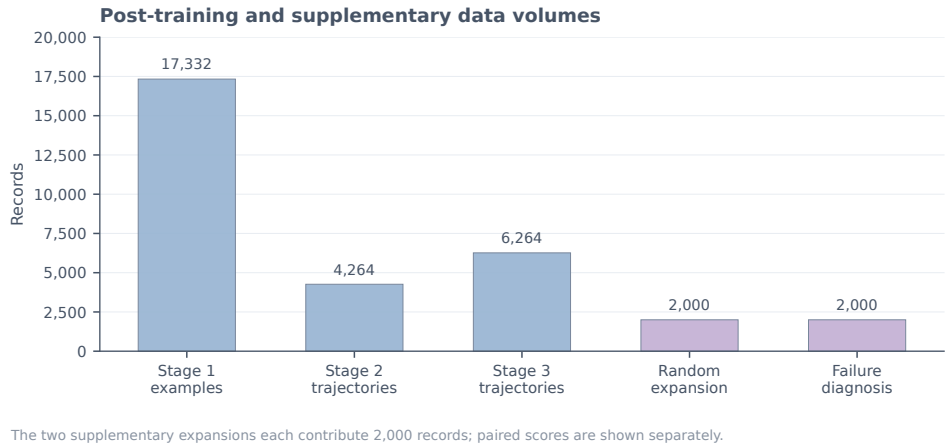


Figure 10: Post-training and supplementary data volumes. Stage 2 and Stage 3 follow the detailed training-configuration tables.

where M_i contains only the recorded model-output positions. Task inputs, tool observations, and external experience are context. Acceptance records are used to filter examples and do not add tokens to the solving sequence.

A compact representation of an accepted sample is:

```
{
  "id": "task-id__trajectory-id",
  "source": {"task_id": "cluster-000/paper-000",
    "trajectory_id": "recorded-run-0001",
    "split": "train"},
  "messages": [
    {"role": "system", "content": "original system prompt"},
    {"role": "user", "content": "original task input"},
    {"role": "assistant",
     "reasoning_content": "recorded reasoning",
     "content": null,
     "tool_calls": []},
    {"role": "tool", "tool_call_id": "call-0001",
     "content": "actual tool return"},
    {"role": "assistant",
     "reasoning_content": "recorded continuation",
     "content": "recorded final answer",
     "tool_calls": []}
  ],
```

Table 16: Loss-mask policy used during serialization.

Message or field	Mask	Treatment
System, user, and tool observation	0	Conditioning context only.
Recorded assistant reasoning	1	Supervise only the text present in the source trace.
Recorded assistant tool calls	1	Supervise actual calls and arguments.
Recorded assistant final content	1	Supervise the actual final response.
Quality and checker metadata	–	Post-hoc filtering and analysis only.

```
  "quality": {"accepted": true, "checker_score": 1.0,
    "posthoc_only": true}
}
```

C.3 Training hyperparameters

The training configuration is summarized in Tables 17–19. Stage 1 adapts the model to the Harness with LoRA, whereas Stages 2 and 3 use full-parameter supervised fine-tuning on scientific trajectories.

Table 17: Stage 1 Harness-adaptation hyperparameters.

Setting	Stage 1: Harness adaptation
Initialization model	Qwen3.6-35B-A3B
Training data	17,332 execution examples
Training method	LoRA SFT
Stage output	Stage 1 adapted checkpoint
Maximum sequence length	65,536 tokens
Optimizer	Fused AdamW
Peak learning rate	1×10^{-5}
Minimum learning rate	1×10^{-6}
Adam (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	1×10^{-8}
Weight decay	0.01
Weight-decay exclusions	Bias and normalization parameters
Learning-rate schedule	Cosine
Warm-up ratio	3%
Global batch size	32
Gradient accumulation steps	8
Maximum training epochs	3
Gradient clipping threshold	1
Computation precision	BF16
Optimizer-state precision	FP32
LoRA rank	32
LoRA alpha	64
LoRA dropout	0.05
Random seed	42

C.4 Failure-guided v1-to-v2 update

BioMaster-A1-v2 starts from v1 and adds targeted task construction, execution collection, and learning-asset updates guided by observed failures. The update targets four recurring capability groups: multi-step reasoning and candidate comparison, precise bioinformatics operations, database semantics, and error recovery with result verification.

Table 18: Stage 2 scientific-trajectory training hyperparameters.

Setting	Stage 2: scientific trajectories
Initialization model	Stage 1 checkpoint
Training data	4,264 scientific trajectories
Training method	Full-parameter SFT
Stage output	BioMaster-A1-v1
Maximum sequence length	131,072 tokens
Optimizer	Distributed Fused AdamW
Peak learning rate	5×10^{-6}
Minimum learning rate	5×10^{-7}
Adam (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	1×10^{-8}
Weight decay	0.01
Weight-decay exclusions	Bias and normalization parameters
Learning-rate schedule	Cosine
Warm-up ratio	5%
Global batch size	32
Gradient accumulation steps	4
Maximum training epochs	3
Gradient clipping threshold	1
Computation precision	BF16
Optimizer-state precision	FP32
LoRA rank	Not applicable
LoRA alpha	Not applicable
LoRA dropout	Not applicable
Random seed	42

Table 19: Stage 3 feedback-guided update hyperparameters.

Setting	Stage 3: feedback-guided update
Initialization model	Stage 1 checkpoint
Training data	6,264 scientific trajectories
Training method	Full-parameter SFT
Stage output	BioMaster-A1-v2
Maximum sequence length	131,072 tokens
Optimizer	Distributed Fused AdamW
Peak learning rate	5×10^{-6}
Minimum learning rate	5×10^{-7}
Adam (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	1×10^{-8}
Weight decay	0.01
Weight-decay exclusions	Bias and normalization parameters
Learning-rate schedule	Cosine
Warm-up ratio	5%
Global batch size	32
Gradient accumulation steps	4
Maximum training epochs	3
Gradient clipping threshold	1
Computation precision	BF16
Optimizer-state precision	FP32
LoRA rank	Not applicable
LoRA alpha	Not applicable
LoRA dropout	Not applicable
Random seed	42

C.4.1 Cluster-level diagnosis and targeted task requirements

Diagnosis is organized by workflow cluster and distinguishes retrieval, planning, execution, and validation failures. A successful tool call is not sufficient: the diagnosis also asks whether the returned evidence was used correctly. Typical cases include failing to compare multiple plausible candidates,

Table 20: Targeted task requirements for failure-guided updates.

Capability target	Required trajectory behavior	Typical checks
Candidate comparison	Generate candidates, collect multi-source evidence, compare and rank, exclude contradictions, validate the preferred candidate	Evidence table or explicit comparison before the final choice.
Precise operations	Expose intermediate calculations under fixed reference, orientation, coordinates, and parameters	Programmatic checks for sequences, coordinates, variants, or numerical outputs.
Database semantics	Map the scientific need to the correct resource, field, and experiment type before querying	Distinguish baseline from differential expression and related field semantics.
Error correction	Retain the error, diagnose it, gather missing evidence or re-call the tool, and verify the correction	Recovery chain remains visible in the recorded interaction.

using the wrong reference genome or strand convention, making a coordinate or numeric error, or selecting a database field that does not match the evidence requested by the question.

Candidate-comparison trajectories follow

$$\begin{aligned}&\text{candidate generation} \rightarrow \text{multi-source evidence} \rightarrow \text{candidate comparison}\\ &\rightarrow \text{contradiction filtering and ranking} \rightarrow \text{priority validation} \rightarrow \text{final answer.}\end{aligned} \tag{13}$$

Precise-operation tasks use programmatic checkers where possible. Database semantics tasks cover Expression Atlas, GEO, Antibodypedia, GWAS Catalog, Open Targets, ClinVar, gnomAD, and Ensembl, and require the trace to retain the reason for choosing a source and field. Error-correction tasks preserve the actual recovery chain; a corrected final answer does not replace that chain.

C.4.2 Supplementary collection and dual-asset update

Targeted tasks use the same eight-way Kimi K3 sampling, 400-call per-run cap, checker validation, group-level score checks, and dual-asset extraction as the initial collection. Accepted executions update the Qwen serialization path and the RU path independently. An RU does not have to be extracted for a trajectory to be eligible for training, and a trajectory and an RU need not be one-to-one.

Training and evaluation snapshots are managed separately. A new model or RU version may be produced during an update, while a model-level comparison keeps the agent configuration, tools, scoring protocol, and RU snapshot fixed within the comparison. The v1-to-v2 result therefore reflects a combined update and does not isolate the effect of LKM targeting from other equal-budget data or training changes.

C.4.3 Random-data and failure-diagnosis expansion

A supplementary experiment adds 2,000 randomly sampled trajectories and 2,000 failure-diagnosis expansions on top of the Stage-1 data. The record volumes are shown in Figure 10; the corresponding component scores are given in Table 21 and Figure 11.

The suffixes d and r denote the failure-diagnosis and random-expansion conditions, respectively. Scores are evaluated on the fixed 203-task component set; no uncertainty estimates are provided for this comparison.

Table 21: Component scores for the supplementary data-expansion comparison.

Model checkpoint	RU disabled	RU enabled
Stage 2 end (BioMaster-A1-v1)	58.95	60.92
BioMaster-A1-v2-d	57.31	61.74
BioMaster-A1-v2-r	59.44	63.71

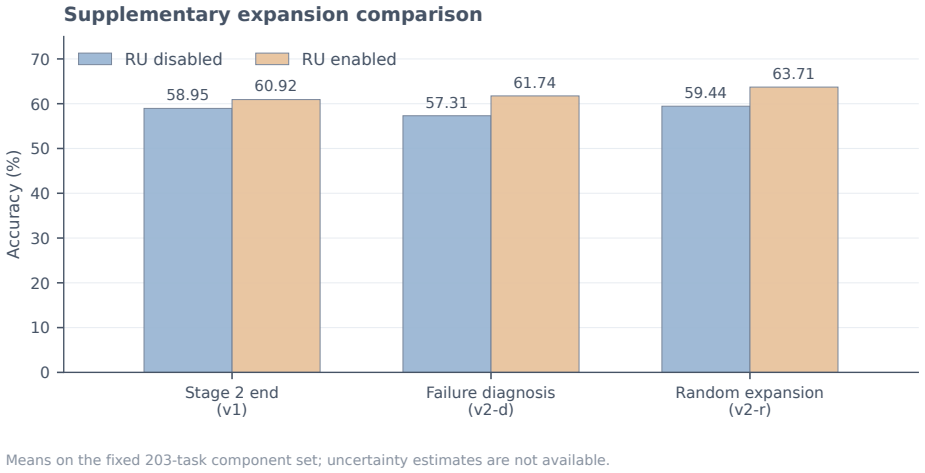


Figure 11: Component scores for the two supplementary data-expansion conditions. Bars show mean scores with and without RU access.

D BENCHMARK PROTOCOLS

D.1 EVALUATION LEVELS AND BENCHMARK SCOPE

We evaluate at two levels. System-level experiments fix Claude Opus 4.6 and compare complete agent configurations. Model-level experiments fix the agent configuration, task inputs, tools, and evaluator and replace only the model or checkpoint. The two levels preserve different native workflows and use different input and score aggregation rules; their scores are reported separately.

All systems and models execute each task or instance three times. We keep all three runs, do not select the best run, do not majority-vote, and do not use an at-least-one-success rule. A repetition is a full solve from task receipt to answer or artifact submission, not a second scoring call on the same answer.

D.2 REPEATED EXECUTION AND SCORE AGGREGATION

For configuration c, benchmark b, and repetition $r \in \{1, 2, 3\}$, let $S_{c,b}^{(r)}$ be the score produced by the benchmark's prescribed parser and scorer. The reported score is

$$\bar{S}_{c,b} = \frac{1}{3} \sum_{r=1}^{3} S_{c,b}^{(r)}. \tag{14}$$

For a binary task-level benchmark with N_b items and correctness $y_{c,b,i}^{(r)} \in \{0, 1\}$,

$$S_{c,b}^{(r)} = \frac{100}{N_b} \sum_{i=1}^{N_b} y_{c,b,i}^{(r)}, \qquad \bar{S}_{c,b} = \frac{100}{3N_b} \sum_{r=1}^{3} \sum_{i=1}^{N_b} y_{c,b,i}^{(r)}. \tag{15}$$

The STELLA task contains a variable number of multiple-choice questions, so its question-level and task-level aggregation is specified separately below.

Table 22: Evaluation hierarchy and benchmark scope.

Level	Fixed factors	Benchmarks	Main comparison
System	Claude Opus 4.6; native agent workflows	STELLA, Biomni-Eval1, BioMystery	Claude Code, STELLA Native, Biomni, BioMaster, and skill-augmented Claude Code.
Model	Agent, tools, inputs, evaluator, and scoring protocol	Same three benchmarks	Qwen3.6-35B-A3B, BioMaster-A1-v1/v2, and reference models.
Component	Tasks, Harness, budgets, scorer; model/RU factors varied by condition	203 LKM tasks	Training, Harness augmentation, and RU access.

Table 23: STELLA direct-MCQ input and output contract.

Component	Contract
Input	Task ID, name, type, description, input resources, success criteria, and all MCQ IDs, stems, and options.
Output	One unambiguous option per question, preferably in the final valid JSON keyed by question ID.
Parsing	Use the last valid submission-format JSON; otherwise use the last explicit question-ID plus single-option mapping.
Question score	Correct option = 1; wrong, missing, illegal, or ambiguous option = 0.
Task score	For task i with m_i questions, $s_{c,i}^{(r)} = m_i^{-1} \sum_j a_{c,i,j}^{(r)}$.

D.3 STELLA Simple Set

The model-level protocol supplies one task's public description and all of its multiple-choice questions in a direct-MCQ input. The 99 tasks contain 308 questions in total, and the same task-level input is executed three times.

The fixed denominator includes unanswered questions. Explanations and tool calls do not add score. The direct-MCQ setting uses a one-hour per-task hard timeout and a concurrency of 20. Platform or network failure may trigger up to two replacement sessions for the same scoring slot; replacements do not create extra samples or allow selection of the highest score. A normal wrong answer or invalid submission is scored under the fixed rule and does not trigger extra solving.

D.4 Biomni-Eval1

Biomni-Eval1 contains 433 instances. Three executions per configuration yield 1,299 scoring slots. The model-level protocol fixes the agent, inputs, tools, and evaluator; the system-level comparison retains each agent's native workflow. The final submission is passed to the fixed parser and evaluator; intermediate discussion and completion claims do not replace the formal answer.

For binary instance score $y_{c,i}^{(r)}$,

$$S_{c,\mathrm{Eval1}}^{(r)} = \frac{100}{433} \sum_{i=1}^{433} y_{c,i}^{(r)}, \qquad \bar{S}_{c,\mathrm{Eval1}} = \frac{100}{1299} \sum_{r=1}^{3} \sum_{i=1}^{433} y_{c,i}^{(r)}. \tag{16}$$

The overall metric is instance-micro accuracy. Refusals, invalid answers, and protocol-defined execution failures remain in the fixed denominator.

D.5 BioMystery v11

BioMystery v11 contains 90 tasks: 73 Human-solvable and 17 Human-hard. Three runs per task give 270 scoring executions, with 219 and 51 executions in the two subsets. The subset labels are fixed by the benchmark and are not reassigned according to a model's answer.

The correctness-only protocol scores the final scientific conclusion against frozen task-specific criteria. Correctness is not string equality with a short answer, and tool success or a complete explanation does not itself imply scientific correctness. For binary task score $y_{c,i}^{(r)}$,

$$S_{c,\mathrm{BioMystery}}^{(r)} = \frac{100}{90} \sum_{i=1}^{90} y_{c,i}^{(r)}. \tag{17}$$

For the two subsets $\mathcal{H}_{\mathrm{solvable}}$ and $\mathcal{H}_{\mathrm{hard}}$,

$$S_{c,\mathrm{solvable}}^{(r)} = \frac{100}{73} \sum_{i \in \mathcal{H}_{\mathrm{solvable}}} y_{c,i}^{(r)}, \tag{18}$$

$$S_{c,\mathrm{hard}}^{(r)} = \frac{100}{17} \sum_{i \in \mathcal{H}_{\mathrm{hard}}} y_{c,i}^{(r)}, \tag{19}$$

$$\bar{S}_{c,\mathrm{BioMystery}} = \frac{73}{90} \bar{S}_{c,\mathrm{solvable}} + \frac{17}{90} \bar{S}_{c,\mathrm{hard}}. \tag{20}$$

Overall is therefore weighted by the number of tasks and is not the simple mean of the two subset accuracies.

D.6 Repeat variability and cross-benchmark summaries

For three complete benchmark scores, the displayed standard deviation is the sample standard deviation:

$$\mathrm{SD}_{c,b} = \sqrt{\frac{1}{2} \sum_{r=1}^{3} \left(S_{c,b}^{(r)} - \bar{S}_{c,b} \right)^2}. \tag{21}$$

This describes variation across full evaluation repeats, not a confidence interval and not training variance. Error-bar overlap is not a significance test. Paired task-level differences require retaining the correspondence between configurations.

The three-benchmark summary is an equal-weight mean of the three benchmark means:

$$S_{c,\mathrm{avg}} = \frac{\bar{S}_{c,\mathrm{STELLA}} + \bar{S}_{c,\mathrm{Eval1}} + \bar{S}_{c,\mathrm{BioMystery}}}{3}. \tag{22}$$

It is not a pooled count across benchmarks and does not include the 203-task component evaluation. A standard deviation for the cross-benchmark mean must be computed from the matched per-repeat cross-benchmark means; it cannot be obtained by averaging three benchmark standard deviations.

D.7 Failures, replacement runs, and software configuration

We distinguish scientific correctness failure from technical non-completion. Wrong answers, explicit refusals, invalid submissions, and protocol-defined model failures remain zeroes in the fixed denominator. A technical run that cannot complete because of a platform or infrastructure fault may be replaced under the applicable benchmark protocol, preserving the original task, configuration, scoring rule, and repetition slot. Replacement does not change the parser or select a better answer.

Table 24: Scoring and replacement rules for invalid submissions and execution failures.

Condition	Treatment
Valid submission satisfying the scoring standard	Score with the prescribed parser and evaluator.
Valid submission with an incorrect answer	Assign score 0.
Explicit refusal, missing answer, or unparseable final submission	Assign score 0 under the protocol and retain the item in the denominator.
Protocol-defined model execution failure	Retain the failure record and assign score 0.
Platform, network, or other technical failure	Apply the replacement rule for the benchmark and configuration; retain the original and replacement records.
Completed evaluation with a low score	Do not rerun or replace the result on the basis of its score.

The system comparison records Claude Code 2.1.177, Biomni 0.0.8, and the frozen STELLA implementation executed with smolagents 1.25.0. Scientific Agent Skills contributes 147 selectable skill entries and BioTender contributes 1,675. Biomni's graph recursion cap is set to 1,000,000 and STELLA's Manager step cap to 200 to avoid internal default truncation; these are different internal units and are not equivalent tool-call budgets. The outer evaluation controller still enforces each task's active termination conditions.

These controls keep task execution, result parsing, and official scoring separate. In particular, the 400-call teacher-trajectory cap in Appendix A.2 is a data-collection setting and is not a universal budget for the three external benchmarks.