EPIGRAPH: META-GRAPH ENGINEERING FOR AGENTIC SCIENTIFIC REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic scientific reasoning is attracting growing attention because it enables agent systems to support scientific decisions and discoveries. Despite rapid progress with increasingly capable large language models, task requirements, intermediate actions, expected observations, and supporting evidence are essential for building capable agent systems in this domain, yet they are often not maintained as a coherent execution structure. This makes execution validity a general and central challenge across scientific tasks, even when the underlying domain knowledge and tools differ. We name this engineering perspective Meta-Graph Engineering and introduce `EpiGraph`, a method that represents scientific execution as a structured graph maintained throughout the task. First, graph operators construct, expand, prune, admit, close, and revise the execution graph; second, typed edges determine whether actions are licensed, what they are expected to produce, and whether their saved results satisfy the corresponding obligations; third, this shared execution structure provides a reusable way to organize agentic scientific reasoning across tasks and systems. We evaluate `EpiGraph` across scientific, mathematical, and biomedical benchmarks with multiple agent configurations and find consistent outcome improvements. Notably, it surpasses biomedical agent systems designed by human experts on BiomniBench without supplying biological prior knowledge or task-specific biological interpretations during execution. These results suggest that `EpiGraph` provides a general execution layer for making agentic scientific reasoning more reliable, inspectable, and reusable.

1 INTRODUCTION

Agentic scientific reasoning is attracting growing attention because agent systems can inspect data and software state, call tools, and carry out multi-step procedures that support scientific decisions and discoveries (Yao et al., 2023b; Schick et al., 2023; Wang et al., 2023a). Scientific tasks make this capability especially demanding: a conclusion must be supported by the data analyzed, the procedure applied, and the evidence preserved during execution (Huang et al., 2025; Rein et al., 2024). In a multi-step analysis, an incorrect cohort definition, an unstated decision rule, or a missing statistical check can propagate into the final conclusion, while the final answer alone does not reveal where the execution became unsupported (Mirza et al., 2025; Mitchener et al., 2025; Wang et al., 2026c; Qu et al., 2026; Wang et al., 2024; Pan et al., 2025). These requirements connect a final conclusion to the analysis unit, decision rule, and supporting artifact that produced it; statistical analyses likewise require an explicit model or comparison rule and, when multiple hypotheses are tested, a stated correction procedure (Cox, 1972; Benjamini & Hochberg, 1995).

Existing agent systems improve scientific work through better prompting, context selection, tool interfaces, and feedback loops, allowing models to access more information, invoke additional capabilities, and revise intermediate outputs (Schulhoff et al., 2024; Khattab et al., 2023; Yang et al., 2024; Madaan et al., 2023). Graph-based systems make other structures explicit, including intermediate reasoning states, agent communication, workflow organization, model selection, and experiential memory (Besta et al., 2023; Yun et al., 2026; Wang et al., 2026b; Yu et al., 2026; Feng et al., 2026a; Bei et al., 2025). These approaches improve what an agent can observe, invoke, or revise, but they do not provide a shared execution representation that connects the conditions supporting an action to the observation it should produce and the evidence that completes its consequence. A general engineering layer is therefore needed to organize these dependencies across scientific tasks,

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.

models, and harnesses. We call this design perspective Meta-Graph Engineering: an execution layer that represents the dependencies among task conditions, agent actions, expected observations, and persisted evidence independently of the domain content, host model, or execution harness. The meta-graph makes these dependencies available while a task is being carried out, so that an agent can determine which actions are supported, what each action should produce, and when a result has enough evidence to be accepted.

Based on this perspective, we introduce `EpiGraph`, a method that constructs and maintains such execution meta-graphs throughout agentic scientific reasoning. `EpiGraph` is organized around three claims. First, legality lives on the edges: prerequisites, expected observations, and persisted receipts are first-class graph nodes, and an action is admitted through typed relations that connect these nodes to the action and to the evidence that discharges its obligation. Second, the graph sits above the task: six domain-independent operators construct and maintain the execution graph with semantics independent of the domain, harness, or model. Third, scientific reasoning provides the most demanding setting for this abstraction, because its receipts include statistical tests, cohort counts, corrected thresholds, and saved analysis artifacts, while the validity of many obligations can be checked by executable predicates rather than left to the model's final judgment. We evaluate `EpiGraph` across scientific, mathematical, condensed-matter physics, and biomedical benchmarks under multiple agent configurations. On the biomedical benchmark, adding `EpiGraph` to a general-purpose agent exceeds the aggregate performance of a biomedical agent architecture designed by human experts, without adding biomedical-specific knowledge or a biomedical-specific harness. Our contributions are:

- We introduce `EpiGraph`, a general-purpose Meta-Graph Engineering method for representing and maintaining execution dependencies across scientific tasks, models, and harnesses.
- We formulate execution legality as typed dependencies among prerequisites, actions, expected observations, and persisted receipts, so that an action is admitted only when its conditions are supported and its resulting evidence can discharge the corresponding obligation.
- We define six domain-independent operators that construct, update, check, and locally repair execution meta-graphs throughout a task.
- We show that, on a biomedical benchmark, adding `EpiGraph` to a general-purpose agent can exceed the aggregate performance of a biomedical agent architecture designed by human experts, without adding biomedical-specific knowledge or a biomedical-specific harness.

2 Related Work

Agentic scientific reasoning. Agentic scientific reasoning studies how language-model agents inspect scientific data, use domain tools, and carry out multi-step analyses whose conclusions remain connected to the procedures and evidence that produced them (Huang et al., 2025; Alber et al., 2026; Xu et al., 2026a; Daghyani et al., 2026). The application surface is broader than a single biomedical workflow: literature agents retrieve and synthesize evidence across full-text papers (Lála et al., 2023), chemistry agents combine language reasoning with reaction, property, and synthesis tools (Bran et al., 2024), autonomous laboratory systems translate plans into instrument actions and experimental measurements (Boiko et al., 2023), and open-ended research systems propose hypotheses, write and run experiments, and generate papers in an iterative loop (Lu et al., 2024). Scientific agents and evaluations also span computational biology, genomics discovery, echocardiography, chemistry, condensed-matter theory, and broader scientific reasoning (Mirza et al., 2025; Wang et al., 2024; Rein et al., 2024; Pan et al., 2025; Qu et al., 2026; Mitchener et al., 2025). These settings expose a common requirement: a scientific agent must preserve the task scope, analysis unit, decision rule, intermediate observations, and supporting artifacts throughout execution, as reflected in process-level and trajectory-based evaluation protocols (Qu et al., 2026; Mitchener et al., 2025; Wang et al., 2026c). Existing work provides increasingly capable models, domain tools, and evaluation protocols for scientific tasks (Huang et al., 2025; Mirza et al., 2025; Alber et al., 2026; Boiko et al., 2023; Lu et al., 2024), but it does not make the dependencies that license an action and complete its consequence a shared execution object. `EpiGraph` addresses this gap by representing those dependencies explicitly and maintaining them throughout the scientific task.

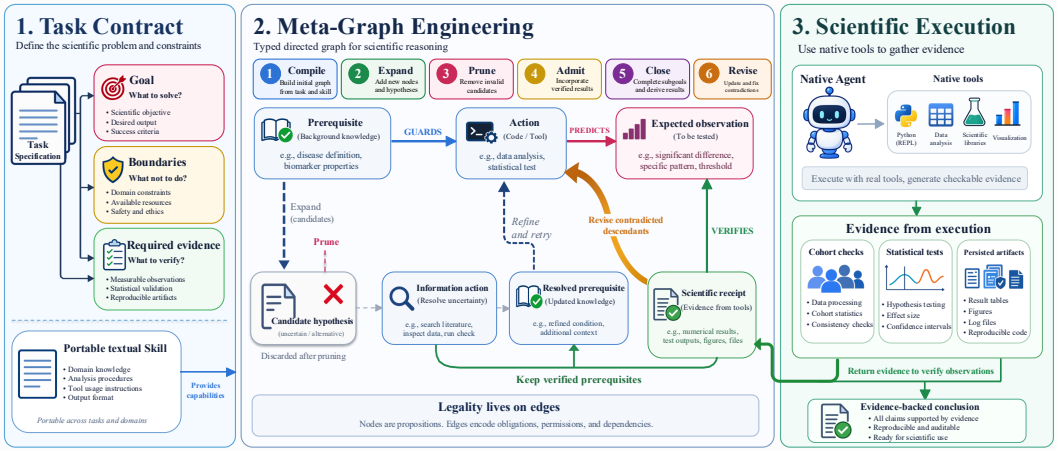


Figure 1: Overview of EpiGraph. A task contract instantiates an execution meta-graph with named node classes and labeled edges whose prerequisite, expected-observation, and receipt nodes are connected by Guards, Predicts, and Verifies; the six operators maintain the graph as the host executes actions and records evidence.

Graph engineering. Graph engineering studies how relationships among reasoning states, agents, tasks, and memory can be made explicit and manipulated during execution. The literature covers graph search and transformation over intermediate thoughts (Yao et al., 2023a; Besta et al., 2023), communication and delegation among agents (Yun et al., 2026), workflow construction and scheduling (Wang et al., 2026b; Yu et al., 2025), and graph memory or routing for selecting models and roles (Feng et al., 2026a). Surveys organize these mechanisms across planning, execution, memory, coordination, verification, provenance, and recovery (Bei et al., 2025; Feng et al., 2026b). They show how graph structure can expose alternatives, combine partial results, route work, or preserve state. Their operational objects include reasoning states, messages, workflows, and component choices. EpiGraph makes execution validity the engineering object: a task-specific graph records what must support an action, what observation it commits to produce, and what persisted evidence can complete its consequence. The appendix traces the broader engineering progression from prompt and context design through harness and loop engineering to graph-based execution representations.

3 EpiGraph: Meta-Graph Engineering

EpiGraph is a meta-graph policy for making execution dependencies explicit while leaving the host model, tools, and harness unchanged. A task instantiates one branching execution graph whose nodes record prerequisites, actions, expected observations, and receipts. The graph is maintained above the task and model: its contents vary with the task, while the policy gives the same meaning to dependencies and evidence across domains.

The graph separates three questions that are often implicit in an agent trajectory. A prerequisite-to-action relation (Guards) records what must hold before an action is admitted; an action-to-expectation relation (Predicts) records what the action is expected to produce; and an evidence-to-expectation relation (Verifies) records which persisted receipt can close that expectation. The policy therefore connects an action to both its enabling conditions and its acceptance evidence. A receipt may be an artifact, a recomputation, a statistical check, or another domain-specific record that can be inspected by the evaluator.

We use a task contract to keep the execution target, boundary, and evidence requirements explicit. Writing the contract as $C = (G, B, R)$ is enough for the main execution loop: G describes the requested terminal state and deliverables, B describes what may be inspected or changed, and R describes the evidence needed to accept the result. The working graph at step t is represented at the level of its core node classes as

$$\mathcal{G}_t = (V_t, E_t), \qquad V_t = P_t \dot{\cup} A_t \dot{\cup} O_t \dot{\cup} R_t, \qquad E_t \subseteq V_t \times \mathcal{K}_{\text{EpiGraph}} \times V_t, \tag{1}$$

where P_t, A_t, O_t, and R_t denote prerequisites, actions, expected observations, and receipts. The graph can also carry uncertainty and decision annotations; these guide the next expansion without changing the four core classes.

For an action a_t, the three edge types, `Guards`, `Predicts`, and `Verifies`, describe a compact admission obligation. In the main text we write this schematically as

$$\mathrm{EdgeReady}_t(a_t) = \mathrm{GuardReady}_t(a_t) \wedge \mathrm{PredictsDeclared}_t(a_t) \wedge \mathrm{ReceiptSpecified}_t(a_t). \quad (2)$$

The first term checks that at least one supported prerequisite path is available, the second records the observation expected from the action, and the third records what evidence will be inspected after execution. This planning condition permits multiple task paths: support sets may be conjunctive, and alternative sets may provide different compliant paths. Information actions that resolve an uncertainty are kept distinct from world actions that change the task state, while both use the same dependency representation.

The graph is maintained through one construction-and-maintenance loop. First, `Compile` identifies the contract and the obligations that matter for the requested outcome. `Expand` makes candidate explanations, information checks, and world actions explicit. `Prune` removes a candidate only when the current compliant continuations do not depend on it. `Admit` then exposes an edge-ready action to the host. After the host returns, `Close` compares the observation with its declared expectation and attaches the saved receipt. If the observation contradicts a prerequisite or fails the declared check, `Revise` marks the affected descendants and opens a local continuation for repair. A branch can consequently finish with a positive, negative, or null scientific result; closure concerns whether the declared analysis and evidence requirements were met.

At the level of a single step, the update is summarized by

$$\mathcal{G}_{t+1} = o_t(\mathcal{G}_t, s_t, o_t^{\mathrm{obs}}, e_t), \qquad o_t \in \{\texttt{Compile}, \texttt{Expand}, \texttt{Prune}, \texttt{Admit}, \texttt{Close}, \texttt{Revise}\}. \quad (3)$$

The operator is selected from the current obligations and returned evidence, so the six operations form a state-dependent loop. A successful local branch contributes to completion when the contract-level condition

$$\mathrm{Complete}_C(\tau_T, s_T, \mathcal{E}_T) = G(s_T) \wedge B(\tau_T) \wedge R(\mathcal{E}_T) \quad (4)$$

holds. The detailed definitions of support, legality, necessity, acceptance, invalidation, and the exact pseudocode for this update are deferred to Appendix B.

At run time, the host executes the admitted action and returns its observation and saved evidence to the policy. The next graph state is then chosen from the updated dependencies. This makes the central object of `EpiGraph` the link between an action, the conditions that license it, and the evidence that accepts its result. The complete node and edge vocabulary, notation table, legality and necessity predicates, support sets, graph-state transitions, `Close` and `Revise` rules, and execution pseudocode are given in Appendix B; the executable policy used by the agents is reproduced in Appendix H. The appendix uses the same node names, edge labels, operator names, and color vocabulary as the main-text overview, so the displayed graph and the formal policy refer to one shared execution object.

4 Experiments

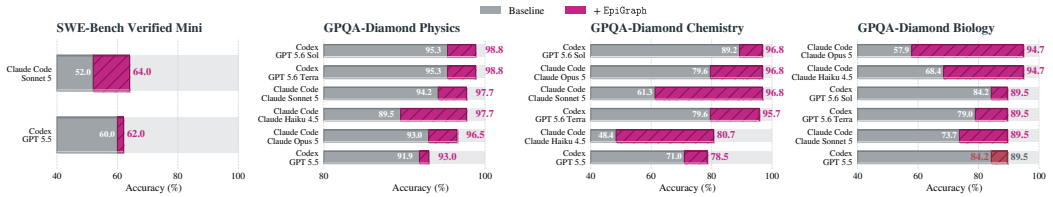
The experiments follow four research questions. **RQ1 (outcome-level):** Does `EpiGraph` improve final task outcomes across agentic scientific reasoning benchmarks and a coding-agent transfer setting, when the task answer or repository state is the evaluation target? **RQ2 (process-level):** Does the same graph organization improve the completeness of a multi-stage scientific workflow, so that its data products, analysis choices, statistical checks, interpretations, and sources remain connected to the final report? **RQ3 (module effectiveness):** Do the three graph edges—`Guards`, `Predicts`, and `Verifies`—and the six operators—`Compile`, `Expand`, `Prune`, `Admit`, `Close`, and `Revise`—each change the resulting execution? **RQ4 (graph intervention consequences):** When a premise, receipt, or label is changed in a recorded graph, which dependent results are reopened or repaired, and which independent results remain valid?

Setup. Each campaign records its benchmark, task set, model, reasoning effort, harness, tools, and evaluator. Every host is evaluated in a baseline and a +`EpiGraph` condition, with persisted answer artifacts and evaluator outputs used for aggregate and task-level results. Each task is run once under

Table 1: Scientific and mathematics datasets: accuracy and execution time on GPQA-Diamond and five mathematics benchmarks. Accuracy is in percent and time is mean per-task wall-clock seconds. The shaded row reports accuracy changes in pp and relative time reductions in percent; upward arrows indicate improvement. All models use medium reasoning effort.

	Methods	Setting	GPQA-Diamond (198)		AIME 2024 (30)		AIME 2025 (30)		BeyondAIME (100)		HMMT 2025 (30)		HMMT 2026 (33)	
			Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
Codex	GPT 5.5	-	81.82	66.66	90.00	58.41	83.33	95.59	69.00	154.76	86.67	103.74	75.76	151.44
		+EpiGraph	**85.35**	**72.56**	**100.00**	**56.78**	**100.00**	**68.20**	**89.00**	**128.57**	**96.67**	**105.99**	**96.97**	**152.21**
		Δ	↑ 3.53 pp	↓ 8.85%	↑ 10.00 pp	↑ 2.79%	↑ 16.67 pp	↑ 28.65%	↑ 20.00 pp	↑ 16.93%	↑ 10.00 pp	↓ 2.17%	↑ 21.21 pp	↓ 0.51%
	GPT 5.6 Sol	-	91.41	246.95	93.33	66.40	93.33	67.96	75.00	88.55	96.67	85.31	93.94	116.04
		+EpiGraph	**96.97**	**158.60**	**100.00**	**66.04**	**100.00**	**69.99**	**93.00**	**117.97**	**100.00**	**91.26**	**100.00**	**96.14**
		Δ	↑ 5.56 pp	↑ 35.78%	↑ 6.67 pp	↑ 0.54%	↑ 6.67 pp	↓ 2.99%	↑ 18.00 pp	↓ 33.21%	↑ 3.33 pp	↓ 6.97%	↑ 6.06 pp	↑ 17.15%
	GPT 5.6 Terra	-	86.36	189.44	96.67	53.32	70.00	97.17	74.00	80.14	96.67	76.59	84.85	90.44
		+EpiGraph	**96.46**	**101.33**	**96.67**	**53.77**	**100.00**	**46.68**	**83.00**	**87.57**	**100.00**	**83.45**	**96.97**	**68.68**
		Δ	↑ 10.10 pp	↑ 46.51%	0.00 pp	↓ 0.84%	↑ 30.00 pp	↑ 51.96%	↑ 9.00 pp	↓ 9.27%	↑ 3.33 pp	↓ 8.96%	↑ 12.12 pp	↑ 24.06%
Claude Code	Claude Opus 5	-	83.33	40.60	73.33	59.54	70.00	119.38	55.00	121.70	73.33	82.58	63.64	89.29
		+EpiGraph	**96.46**	**40.04**	**93.33**	**46.37**	**93.33**	**54.52**	**66.00**	**101.14**	**90.00**	**71.06**	**87.88**	**75.49**
		Δ	↑ 13.13 pp	↑ 1.38%	↑ 20.00 pp	↑ 22.12%	↑ 23.33 pp	↑ 54.33%	↑ 11.00 pp	↑ 16.89%	↑ 16.67 pp	↑ 13.95%	↑ 24.24 pp	↑ 15.46%
	Claude Sonnet 5	-	76.77	30.23	83.33	70.59	80.00	84.45	43.00	104.06	50.00	103.47	60.61	76.16
		+EpiGraph	**96.46**	**27.87**	**93.33**	**57.31**	**86.67**	**61.01**	**63.00**	**107.96**	**70.00**	**54.32**	**69.70**	**77.80**
		Δ	↑ 19.69 pp	↑ 7.81%	↑ 10.00 pp	↑ 18.81%	↑ 6.67 pp	↑ 27.77%	↑ 20.00 pp	↓ 3.74%	↑ 20.00 pp	↑ 47.50%	↑ 9.09 pp	↓ 2.15%
	Claude Haiku 4.5	-	68.18	110.71	83.33	86.96	70.00	104.80	48.00	135.39	53.33	104.40	45.45	101.48
		+EpiGraph	**89.39**	**94.04**	**90.00**	**75.95**	**90.00**	**98.58**	**60.00**	**121.22**	**73.33**	**108.52**	**72.73**	**87.48**
		Δ	↑ 21.21 pp	↑ 15.06%	↑ 6.67 pp	↑ 12.66%	↑ 20.00 pp	↑ 5.93%	↑ 12.00 pp	↑ 10.47%	↑ 20.00 pp	↓ 3.95%	↑ 27.28 pp	↑ 13.80%

Accuracy by domain and examination split



Gray bars are baselines; hatched magenta bars are +EpiGraph.

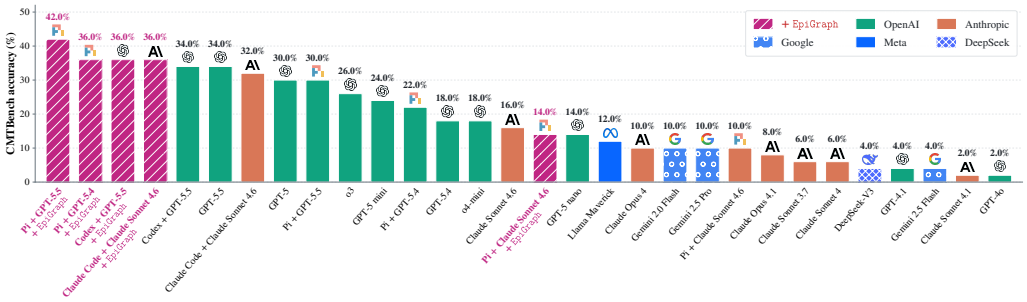


Figure 3: Overall CMT-Bench accuracy across the recorded systems, with the EpiGraph configurations highlighted.

each condition; API errors trigger request-level retries. BiomniBench-DA assigns one hour to each task; the other campaigns use time budgets sufficient for every registered task to return a result. Tables denote the baseline and +EpiGraph conditions by - and +EpiGraph, respectively.

4.1 RQ1: Outcome-Level Evaluation

RQ1 asks whether EpiGraph improves tasks whose evaluation target is a direct answer or a final repository state. We evaluate matched baseline and +EpiGraph conditions on scientific question answering, competition mathematics, condensed-matter theory, and coding-agent transfer, using each benchmark's native evaluator. Table 1 shows a broad outcome-level effect: accuracy increases in 35 of 36 model–dataset pairs and is unchanged in the remaining pair. The same pattern appears in the GPQA domain split, the mathematics suites, and the condensed-matter comparison in Figure 3; the magnitude varies with the task family and host configuration.

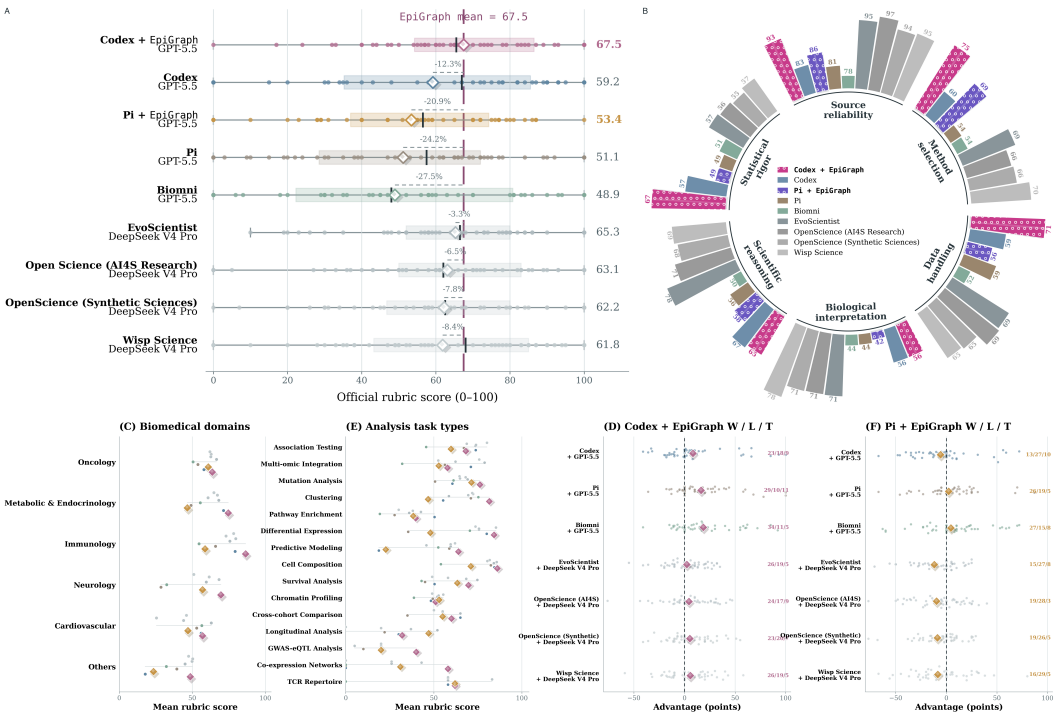


Figure 4: BiomniBench-DA overview across outcome level, rubric dimensions, biomedical domains, analysis task types, and paired task changes. The upper row contains (A) score distributions for the nine systems and (B) the six rubric dimensions. The lower row contains (C) domain-level means, (E) analysis-type means, (D) task-wise changes for Codex + `EpiGraph` against the seven non-`EpiGraph` baselines, and (F) the corresponding changes for Pi + `EpiGraph`. In the W/L/T panels, each row is one baseline; points retain the baseline's color, while the large diamond marks the mean advantage of the corresponding `EpiGraph` configuration.

The outcome gain is accompanied by configuration-dependent time changes. For example, GPT 5.5 gains 20 percentage points on BeyondAIME while reducing mean time by 16.93%, whereas Claude Sonnet 5 gains the same accuracy margin with a 3.74% increase. Under Pi, GPT 5.5 improves from 30% to 42% on CMT-Bench, while the native Codex and Claude Code comparisons improve by 2 and 4 percentage points. Thus the outcome-level evidence spans several answer formats and harnesses, with the resource change depending on the execution setting. The complete domain, examination, and coding-agent breakdowns are in Appendix C and Appendix D.2.

4.2 RQ2: Process-Level Evaluation

RQ2 asks whether `EpiGraph` improves the completeness and state consistency of a multi-stage scientific analysis, and whether a general-purpose agent equipped with `EpiGraph` can exceed specialized scientific-agent configurations in the same evaluation. We use the 50-task BiomniBench-DA data-analysis subset (Qu et al., 2026). Each task supplies a dataset and research question, and its expert-authored rubric scores the recorded trajectory and final answer for data handling, cohort construction, method choice, statistical evidence, interpretation, and source-grounded reporting. Gemini 3.1 Pro assigns partial credit on these criteria and aggregates them into a 0–100 score, so the evaluation measures whether intermediate products and decisions remain connected to the final report.

The aggregate comparison establishes the main process-level result. Codex scores 59.16, below the self-evolving EvoScientist at 65.26, OpenScience (AI4S Research) at 63.10, OpenScience (Synthetic Sciences) at 62.20, and Wisp Science at 61.80. Codex alone therefore trails every one of these specialized scientific-agent configurations. Codex + `EpiGraph` scores 67.48, the highest mean among the nine systems in Figure 4: adding `EpiGraph` reverses that ranking, with gains of 2.22 points over EvoScientist, 4.38 and 5.28 points over the two OpenScience configurations, 5.68 points over Wisp Science, and 18.56 points over the biomedical Biomni agent (48.92). Figure 11 shows

how the gain is distributed over tasks: non-zero scores increase from 43 to 49, tasks reaching 40 points from 36 to 45, and full-credit tasks from 5 to 9.

Panel (B) of Figure 4 identifies where the process-level gain comes from. Source reliability rises from 82.69 to 93.27, data handling from 59.34 to 70.88, method selection from 60.29 to 75.00, and statistical rigor from 57.21 to 66.67. Biological interpretation remains 56.45, while scientific reasoning changes from 66.67 to 65.28. The gain is therefore concentrated on the obligations that connect data products, analysis choices, checks, and saved sources, rather than on biological interpretation itself. The run uses a general-purpose coding agent and adds no biological interpretation, biomedical tool workflow, or biomedical-specific harness; its process-level improvement comes from the execution organization supplied by `EpiGraph`.

Panels (C)–(F) reveal a structural pattern rather than a uniform score increase. The largest gains occur in predictive modeling, clustering, survival analysis, and mutation analysis. These workflows are organized around explicit cohorts, labels, quantitative outputs, and decision rules: clustering depends on a defined feature matrix and group assignment, predictive modeling on a trainable outcome and held-out evaluation, survival analysis on cohort and censoring rules, and mutation analysis on reproducible filters and counts. These are precisely the dependencies that `EpiGraph` keeps connected from prerequisite to action, expected observation, and saved receipt. The resulting improvements are especially visible in general biology and neurology, where Codex rises from 17.8 to 48.6 and from 56.8 to 69.5, and in immunology, where it reaches 86.4.

The smaller gains in pathway enrichment, multi-omic integration, and association testing have a different interpretation. These tasks rely more heavily on mapping measurements to biological mechanisms, selecting meaningful hypotheses, and explaining why a molecular relationship matters; execution structure can preserve the cohort, contrast, statistical test, and evidence, but it cannot supply that biological interpretation. Even in these settings, `EpiGraph` improves the organization of the analysis and its supporting receipts, which helps explain why the aggregate remains positive while the largest gains occur in more explicitly structured workflows. Panel (D) records 23/9/18 wins/ties/losses against Codex, 26/5/19 against EvoScientist, and 34/5/11 against Biomni. Panel (F) shows transfer to Pi, where Pi + `EpiGraph` reaches 53.38 versus 51.14 for Pi. Figure 5 gives the same result by classification: Panel (a) shows broad domain coverage, with `EpiGraph` strongest in immunology and metabolic analysis, while Panel (b) shows its clearest task-level advantages in clustering, differential expression, predictive modeling, and mutation analysis; EvoScientist remains stronger on multi-omic integration and survival analysis. Taken together, these panels show what the graph contributes: it captures a domain-general execution structure that raises process-level performance across biomedical workflows, while the remaining variation follows how much a task depends on biological explanation beyond procedural and evidential organization. For the rubric protocol, task taxonomy, threshold curves, loss analysis, and case-level artifacts, see Appendix E and Appendix E.5.

4.3 RQ3: Module-Level Effectiveness

RQ3 asks which parts of the `EpiGraph` policy change task outcomes or execution behavior. We run matched ablations on the held-out GPQA-Diamond and BeyondAIME tasks, keeping task IDs, models, reasoning effort, harness, evaluator, and execution setting fixed. The variants remove one typed dependency, one of the six operators, or the graph structure while retaining an aligned checklist of the same task-facing requirements, following the plan-and-execute and reason–act comparison points used in prior agent policies (Wang et al., 2023b; Yao et al., 2023b).

Figure 6 reports every variant relative to +`EpiGraph`. Removing `Guards`, `Predicts`, or `Verifies` changes the corresponding prerequisite, expected-observation, or receipt dependency. Removing any of the six operators tests the graph update that it performs; the observed changes appear in both accuracy and execution time. For example, with GPT 5.6 Terra the full condition reaches 96.46% on GPQA and 83.00% on BeyondAIME, compared with 78.79%/63.00% without `Admit` and 85.35%/68.00% for the aligned checklist.

The checklist retains the same task-facing requirements in an ordered list but removes the typed dependencies between them. Its lower position than the full graph in the matched map gives the direct comparison requested by checklist-based agent policies: listing obligations helps, while con-

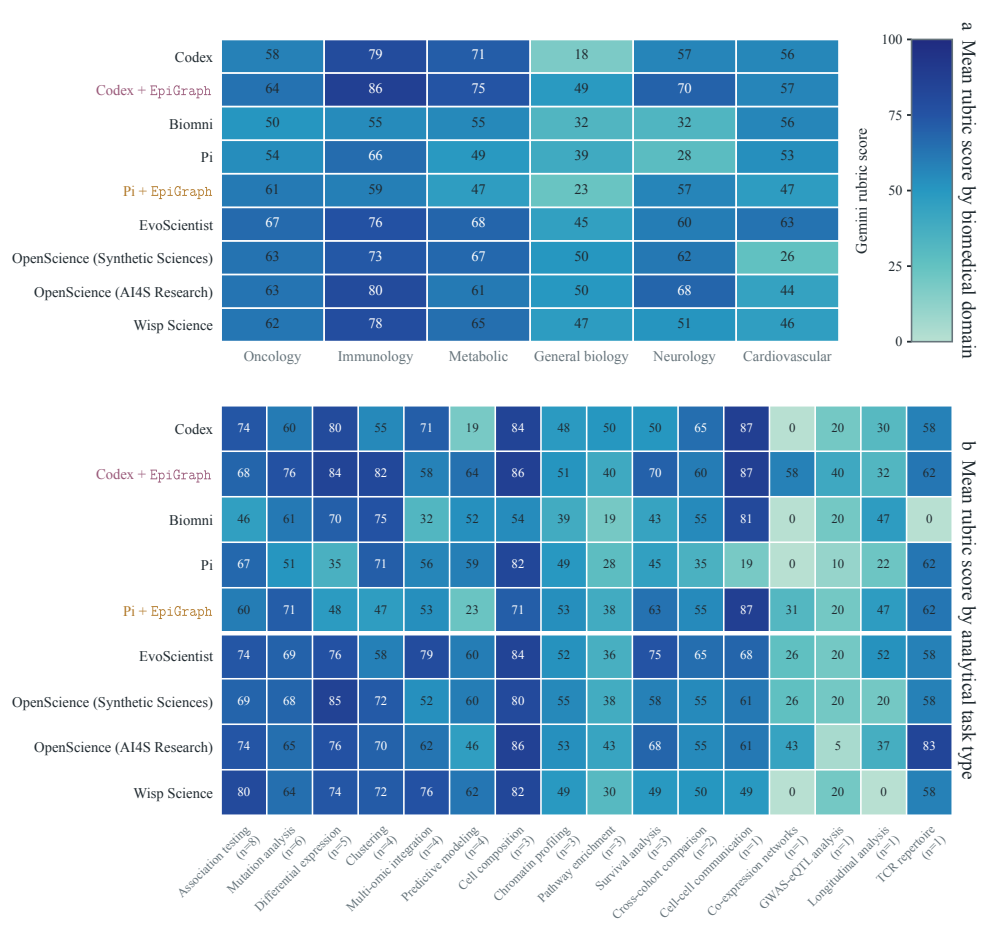


Figure 5: BiomniBench-DA performance under two classification views. Panel (a) reports mean Gemini rubric scores grouped by biomedical domain, with the shared score colorbar immediately to its right; panel (b) reports the same scores grouped by analytical task type. Values are written into the cells. The appendix gives the corresponding task-count matrix and the six detailed case scores.

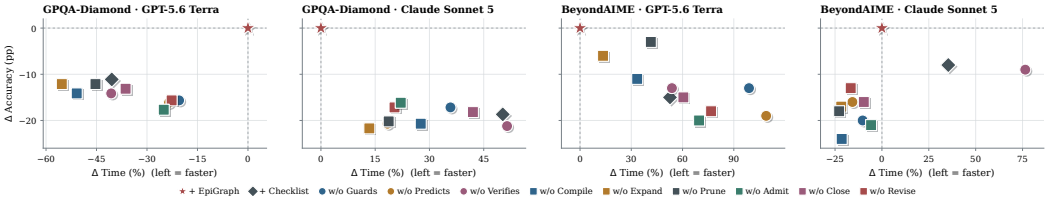


Figure 6: Matched ablation accuracy and time. The display reports accuracy and mean per-task time under matched ablations relative to +EpiGraph; circles denote edge ablations, squares operator ablations, and the diamond the checklist condition.

necting prerequisites, expected observations, and receipts changes which actions remain supported and which outputs are accepted. The condition-level failure inventory is reported in Appendix F.

4.4 RQ4: Graph Intervention Consequences

RQ4 asks whether graph state has the predicted operational consequences for subsequent execution. We use the recorded high-scoring BiomniBench-DA execution in Appendix E.6, whose graph contains patient eligibility, score-derived subgroup rules, denominator sensitivity, patient aggregation,

typed dependencies, and saved receipts. Each intervention targets one kind of graph state: a cohort premise, a verification receipt, or a patient-level label. The expected response is correspondingly specific: premise changes should reopen dependent cohort and aggregation work, receipt deletion should reopen the verification obligation before aggregation closes, and a local label conflict should trigger revision only for the affected branch while preserving independent work. We replay the same trace under these three changes and evaluate every resulting trace with the same rubric.

Starting from the recorded `da-14-3` graph, we replayed the three variants summarized in Figure 7. The original trace scored 100, the premise mutation scored 66, receipt deletion scored 85, and the label conflict scored 100. The premise mutation reopened its downstream cohort and aggregation descendants and reduced method-selection and scientific-reasoning scores; receipt deletion preserved the final counts while reducing method completeness; and the label conflict triggered the `Revise` operator on the affected descendants while preserving unrelated patients and the final aggregate. All three responses follow the predicted direction of the intervention. The result supports a specific mechanism claim for `EpiGraph`: graph state is an operational dependency layer whose changes propagate to the obligations that depend on that state, while locally independent work remains available. The full fork/commit trace alignment, six-dimensional rubric profile, and claim-level interpretation are given in Appendix G; the remaining BiomniBench walkthroughs are provided in Appendix E.5.

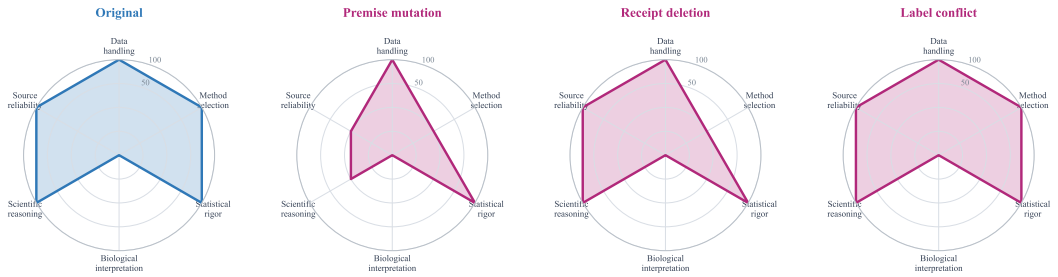


Figure 7: Intervention performance for the recorded `da-14-3` trace under `+ EpiGraph` and three targeted replay variants, evaluated with the same six-dimensional Gemini 3.1 Pro rubric. The four panels are arranged in one row: original trace, premise mutation, receipt deletion, and label conflict. The prose describes the concrete trace edits and their downstream effects; the panels show their resulting criterion profiles.

The remaining BiomniBench walkthroughs are provided in Appendix E.5. They use the same graph vocabulary and visual grammar, with node counts and branch structures following their actual traces. The appendix presents successful, degraded, and revision-heavy execution patterns across these cases.

5 Conclusion

We introduced `EpiGraph`, a meta-graph engineering policy for agentic scientific reasoning. Its six operators—`Compile`, `Expand`, `Prune`, `Admit`, `Close`, and `Revise`—maintain task-indexed graphs in which action legality is carried by `Guards`, `Predicts`, and `Verifies` edges. Across four benchmark families spanning condensed-matter theory, biomedical analysis, science reasoning, and competition mathematics, reported configurations improve with `EpiGraph`. `EpiGraph` therefore establishes meta-graphs as an engineering layer above tasks and models, where scientific actions can be inspected, tested, and locally repaired before their claims are accepted. Most importantly, on the biomedical benchmark, a general-purpose agent equipped with `EpiGraph` exceeds specialized biomedical agent frameworks, including self-evolving systems and architectures designed by human experts, without biomedical-specific knowledge, tools, workflows, or a biomedical-specific harness. This result identifies Meta-Graph Engineering as a general technique for agentic scientific reasoning: maintaining execution dependencies as an explicit graph provides a reusable way to organize and improve scientific work across domains.

REFERENCES

Samuel Alber, Bowen Chen, Eric Sun, Alina Isakova, Aaron J. Wilk, and James Zou. CellVoyager: AI CompBio agent generates new insights by autonomously analyzing biological data. *Nature Methods*, 2026. doi: 10.1038/s41592-026-03029-6.

Yuanchen Bei, Weizhi Zhang, Siwen Wang, Weizhi Chen, Sheng Zhou, Hao Chen, Yong Li, Jiajun Bu, Shirui Pan, Yizhou Yu, Irwin King, Fakhri Karray, and Philip S. Yu. Graphs meet ai agents: Taxonomy, progress, and future opportunities, 2025.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models, 2023. URL https://arxiv.org/abs/2308.09687.

Daniil A. Boiko, Robert MacKnight, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. ChemCrow: Augmenting large-language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B*, 34 (2):187–202, 1972. doi: 10.1111/j.2517-6161.1972.tb00899.x.

Matin Daghyani, Lyuyang Wang, Nima Hashemi, Bassant Medhat, Baraa Abdelsamad, Eros Rojas Velez, XiaoXiao Li, Michael Tsang, Christina Luong, Purang Abolmaesumi, and Teresa Tsang. EchoAgent: Guideline-centric reasoning agent for echocardiography measurement and interpretation. *Journal of the American Society of Echocardiography*, 2026. doi: 10.1016/j.echo.2026.05.023.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. GraphPlanner: Graph memory-augmented agentic routing for multi-agent llms, 2026a.

Yuyuan Feng, Zhishang Xiang, Chaobin Yang, Qichao Ma, Zerui Chen, Yujing Zhang, Ke Huang, Chuanjie Wu, Zhaoxu Liu, Yili Wang, Xin He, et al. Graph engineering in the era of llm agents: From individual intelligence to system intelligence, 2026b.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. PromptBreeder: Self-referential self-improvement via prompt evolution, 2023.

Andrew Gelman and Hal Stern. The difference between "significant" and "not significant" is not itself statistically significant. *The American Statistician*, 60(4):328–331, 2006. doi: 10.1198/000313006X152649.

Richard R. Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 1998. doi: 10.1119/1.18809.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 1969. doi: 10.1145/363235.363259.

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.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines, 2023.

William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. doi: 10.1080/01621459.1952.10483441.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White. PaperQA: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023. URL https://arxiv.org/abs/2312.07559.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

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, pp. 46534–46594, 2023. doi: 10.52202/075280-2019.

A. H. Mirza, Nawaf Alampara, Sreekanth Kunchapu, Martiño Ríos-García, Benedict Emoekabu, Aswanth Krishnan, Tanya Gupta, Mara Schilling-Wilhelmi, Macjonathan Okereke, Anagha Aneesh, Mehrdad Asgari, and Juliane Eberhardt. A framework for evaluating the chemical knowledge and reasoning abilities of large language models against the expertise of chemists. *Nature Chemistry*, 2025. doi: 10.1038/s41557-025-01815-x.

Ludovico Mitchener, Jon M. Laurent, Alex Andonian, Benjamin Tenmann, Siddharth Narayanan, Geemi P. Wellawatte, Andrew White, Lorenzo Sani, and Samuel G. Rodriques. BixBench: A comprehensive benchmark for LLM-based agents in computational biology. *arXiv preprint arXiv:2503.00096*, 2025. URL https://arxiv.org/abs/2503.00096.

Haining Pan, James V. Roggeveen, Erez Berg, Juan Carrasquilla, Debanjan Chowdhury, Surya Ganguli, Federico Ghimenti, Juraj Hasik, Henry Hunt, Hong-Chen Jiang, Mason Kamb, Ying-Jer Kao, Ehsan Khatami, Michael J. Lawler, Di Luo, Titus Neupert, Xiaoliang Qi, Michael P. Brenner, and Eun-Ah Kim. CMT-Benchmark: A Benchmark for Condensed Matter Theory Built by Expert Researchers, 2025. URL https://arxiv.org/abs/2510.05228.

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, and Hai-Tao Zheng. Natural-language agent harnesses, 2026.

Yuanhao Qu, Yingzhou Lu, Xinming Tu, Serena Zhang, Tianwei She, Alexander Glenn Shaw, Jou-Ho Shih, Bingqing Zhao, Minjie Shen, Haochen Yang, Jielin Yan, Rongchuan Zhang, Xinze Wu, Tingting Li, Bin Zhou, Ning Wang, Adam Ma, Le Cong, Xiaobo Hu, Yuan Jiang, Jiayun Dong, Tao Peng, Jure Leskovec, and Kexin Huang. BiomniBench: Process-level Evaluation of LLM Agents for Real-world Biomedical Research. *bioRxiv*, 2026. doi: 10.64898/2026.05.12.724604.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2024. doi: 10.48550/arXiv.2311.12022.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987. doi: 10.1016/0377-0427(87)90125-7.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. The Prompt Report: A Systematic Survey of Prompt Engineering Techniques, 2024. URL https://arxiv.org/abs/2406.06608.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1), 1904. doi: 10.2307/1412159.

The Open Science Desktop Contributors. Open Science Desktop: a local-first, model-agnostic AI research workbench. Zenodo software release, version 0.5.2, 2026. URL https://github.com/ai4s-research/open-science.

Jianing Wang, Linsen Guo, Zhengyu Chen, Qi Guo, Hongyu Zang, Wenjie Shi, Haoxiang Ma, Xiangyu Xi, Xiaoyu Li, Wei Wang, and Xunliang Cai. HeavySkill: Heavy thinking as the inner skill in agentic harness. *arXiv preprint arXiv:2605.02396*, 2026a. URL https://arxiv.org/abs/2605.02396.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A Survey on Large Language Model based Autonomous Agents, 2023a. URL https://arxiv.org/abs/2308.11432.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models, 2023b. URL https://arxiv.org/abs/2305.04091.

Xiang Wang, Zongtao Yang, Zhuojian Hong, Shuhao Zhang, and Wei Wei. FusionFlow: Enabling deep structural exploration for automated agentic workflow generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 27718–27760, 2026b. doi: 10.18653/v1/2026.acl-long.1278.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. SciBench: Evaluating college-level scientific problem-solving abilities of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2307.10635.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, Kai Tian, Zhenzhao Yuan, Jincheng Zhong, Weizhi Wang, Ning Ding, Bowen Zhou, and Kaiyan Zhang. NatureBench: Can coding agents match the published SOTA of nature-family papers? *arXiv preprint arXiv:2606.24530*, 2026c. URL https://arxiv.org/abs/2606.24530.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022. URL https://arxiv.org/abs/2201.11903.

Weize Xu, Erwin Poussi, Quan Zhong, Zehua Zeng, Christopher Zou, Xuehai Wang, Yifan Lu, Miao Cui, Daiji Okamura, Cinlong Huang, Jiayuan Ding, Zhe Zhao, Yuheng Yang, Xinhai Pan, Varshini Vijay, Naoki Konno, Nianping Liu, Lei Li, X. Rosa Ma, Stephanie D. Conley, Colin Kern, William R. Goodyer, Bogdan Bintu, Quan Zhu, Neil C. Chi, Jiang He, Lorenz Rognoni, Xiuwei Zhang, Jun Wu, David Ellison, Marlene Rabinovitch, Jesse M. Engreitz, and Xiaojie Qiu. PantheonOS: An evolvable multi-agent framework for automatic genomics discovery. *bioRxiv*, 2026a. doi: 10.64898/2026.02.26.707870.

Zhougeng Xu et al. xuzhougeng/wisp-science: v1.5.0: Share. Zenodo software release, version 1.5.0, 2026b. URL https://github.com/xuzhougeng/wisp-science.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, 2023a. URL https://arxiv.org/abs/2305.10601.

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*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Junwei Yu, Yepeng Ding, and Hiroyuki Sato. DynTaskMAS: A Dynamic Task Graph-driven Framework for Asynchronous and Parallel LLM-based Multi-Agent Systems. *Proceedings of the International Conference on Automated Planning and Scheduling*, 35:288–296, 2025. doi: 10.1609/icaps.v35i1.36130. URL https://ojs.aaai.org/index.php/ICAPS/article/view/36130.

Zhaochen Yu, Yingcheng Wu, Zhenfei Yin, Kaiyuan Chen, Zhe Zhao, Mengdi Wang, Shuicheng Yan, and Ling Yang. Recursive experiential–working memory evolution for long-horizon agent harnesses. *arXiv preprint arXiv:2608.24876*, 2026. URL https://arxiv.org/abs/2608.24876v1.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-Agents: A graph-based framework for multi-agent llm collaboration, 2026.

Zehua Zeng, Xu Meng, Lei Hu, Chen Li, Peng Liu, Yu Shi, Xuejiao Ma, Lianchong Gao, Xuehai Wang, Zhi Luo, Yawen Zheng, Jieshen Xian, Ziheng Lin, Haoliang Zhu, Zhaorui Jiang, Sheng Mao, Yifan Lu, Wenzhuo Tang, Qiangwei Peng, Yuqing Ma, Liping Zhou, Cencan Xing, Xuegong Zhang, Yuanyan Xiong, and Hongwu Du. OmicOS: A Comprehensive Omics Ecosystem Infrastructure and Agent System for the AI Era. *bioRxiv*, 2026. doi: 10.64898/2026.06.11.731775.

Wentao Zhang, Zhe Zhao, Haibin Wen, Yingcheng Wu, Cankun Guo, Ming Yin, Bo An, and Mengdi Wang. Autogenesis: A self-evolving agent protocol. *arXiv preprint arXiv:2604.15034*, 2026. doi: 10.48550/arXiv.2604.15034.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

APPENDIX CONTENTS

A Related Work . . . 15
- A.1 Agentic Scientific Discovery . . . 15
- A.2 Agent Systems . . . 15

B Formalization Definition of `EpiGraph` . . . 17
- B.1 Execution primitives . . . 17
- B.2 Execution Edge Types . . . 18
- B.3 Six Operators . . . 18
- B.4 Notation and execution policy . . . 21

C Experimental Environment and Protocol . . . 21

D RQ1: Outcome-Level Evaluation . . . 24
- D.1 Outcome-Level Benchmark Breakdown . . . 24
- D.2 Coding-Agent Transfer . . . 25
- D.3 RQ1 Supplement: Outcome-Level Efficiency Measurements . . . 25

E RQ2: Process-Level Biomedical Evaluation . . . 29
- E.1 RQ2 Supplement: Process-Level Rubric and Completion Scores . . . 29
- E.2 Gemini 3.1 Pro Rubric and Scoring Protocol . . . 29
- E.3 Detailed Case Score Atlas . . . 31
- E.4 RQ2 Supplement: Process-Level Boundary Cases . . . 32
- E.5 Concrete Process-Level Case Performance on BiomniBench-DA . . . 33
- E.6 Immune Subgroups: Define the Rule Before Counting . . . 33
- E.7 Metabolomics: A Complete Result Can Contain Zero Discoveries . . . 36
- E.8 Proteomics: Close Both the Set Comparison and the Statistical Test . . . 39
- E.9 Immune Clustering: The Direction of a Relation Is a Prerequisite . . . 43
- E.10 Survival Analysis: Verify the Unit Before Testing the Claim . . . 45
- E.11 Cancer Genomics: Carry Every Calculation into the Scientific Answer . . . 48
- E.12 Evidence Obligations as a Route to Scientific Specialization . . . 51
- E.13 What the Executions Actually Produce . . . 51
- E.14 Why These Graphs Work Across Scientific Tasks . . . 56

F RQ3: Module Ablations and Failure Inventory . . . 57

G RQ4: Prospective Intervention Consequences . . . 58

H Full Execution Policy . . . 60

I Discussion . . . 64

J AI Assistance . . . 64

A Related Work

This appendix expands the two themes compressed in the main-text Related Work.

A.1 Agentic Scientific Discovery

Agentic scientific discovery covers systems that use language models to search, analyze, and act on scientific information or experimental resources. Literature agents retrieve and synthesize evidence across full-text papers, as in PaperQA and its LitQA evaluation (Lála et al., 2023). Chemistry agents connect language reasoning to reaction, property, and synthesis tools; ChemCrow plans and executes chemistry workflows across synthesis, drug discovery, and materials design (Bran et al., 2024). Autonomous laboratory systems extend this loop into physical experimentation: Coscientist translates natural-language objectives into instrument operations, measurements, and subsequent decisions (Boiko et al., 2023). Open-ended research systems broaden the scope again by proposing hypotheses, writing and running experiments, analyzing their results, and producing scientific reports in an iterative cycle (Lu et al., 2024). Biomedical and computational-biology agents apply the same idea to data analysis, genomics, and domain-specific interpretation (Huang et al., 2025; Alber et al., 2026; Xu et al., 2026a), while specialized systems target clinical measurement and interpretation (Daghyani et al., 2026). Together these applications show the growing range of scientific activities that can be delegated to agents. They also make the execution problem concrete: each activity links a task scope and analysis unit to intermediate observations, decisions, and saved evidence. Process-level and trajectory-based evaluations make this link explicit by scoring not only a final answer but also the completeness of the work that produced it (Qu et al., 2026; Mitchener et al., 2025; Wang et al., 2026c).

A.2 Agent Systems

The following engineering layers describe how agent systems are specified, provided with information, connected to tools, run through feedback loops, and organized as graphs. The progression follows the engineering perspective of (Feng et al., 2026b); each layer changes a different control surface around an agent, while a complete system may combine them.

Prompt Engineering. Prompt engineering controls instructions, demonstrations, and response structure within a model invocation (Schulhoff et al., 2024). Chain-of-thought exposes intermediate steps, while Plan-and-Solve separates planning from execution (Wei et al., 2022; Wang et al., 2023b). PromptBreeder searches task prompts and the mutation prompts that improve them, while DSPy replaces hand-written prompt strings with declarative, optimizable modules (Fernando et al., 2023; Khattab et al., 2023). This layer changes how a model maps its immediate input to an output. The meta-graph records whether an action is licensed by the current world state.

Context Engineering. Context engineering controls which evidence, memory, and prior experience are available when a decision is made. ExpeL extracts reusable natural-language lessons and retrieves prior experiences at inference time, illustrating how context selection can improve behavior without parameter updates (Zhao et al., 2024). This layer can supply a missing fact to `EpiGraph`, but the meta-graph separately records whether that fact guards an action and whether later evidence closes the resulting branch. The distinction becomes concrete when a retrieved analysis contains a useful cohort definition. Retrieval makes the definition available; admission still requires checking that the current dataset satisfies it. The planning, execution, memory, and coordination taxonomy of (Bei et al., 2025) further shows why graph-structured memory is only one use of graphs in agents. A memory edge can help locate relevant experience without serving as evidence that a current execution obligation has been discharged.

Harness Engineering. Harness engineering controls the execution substrate around a model: tools, interfaces, state transfer, validation, and lifecycle. SWE-agent demonstrates that an agent–computer interface changes software-engineering performance, while Natural-Language Agent Harnesses makes run-level policy editable and analyzable (Yang et al., 2024; Pan et al., 2026). Recent orchestration systems and Autogenesis further alter inference-time coordination and agent protocols (Wang et al., 2026a; Zhang et al., 2026). `EpiGraph` places admissibility conditions in the meta-graph while the harness executes actions. This interface matters for portability: changing a file API or a tool schema changes how an action is executed, whereas checking a prerequisite before that

action remains the same control operation. In biomedical systems, Biomni integrates tools and resources for research workflows (Huang et al., 2025), and OmicOS emphasizes state-aware capability contracts (Zeng et al., 2026). These are close operational connections: `EpiGraph` brings prerequisites, expected observations, and resulting evidence into a shared dependency representation that also supports the `Prune` and `Revise` operators.

Loop Engineering. Loop engineering controls repeated interaction with feedback. Reflexion stores linguistic feedback in episodic memory to guide subsequent trials (Shinn et al., 2023). ReAct alternates reasoning and action, Self-Refine iterates feedback and revision, and ExpeL accumulates experience across tasks (Yao et al., 2023b; Madaan et al., 2023; Zhao et al., 2024). Recuris links task state, capability invocation, action, and observation in a structured trace; a fixed Meta-Agent uses that trace to localize changes to experiential memory, the working-memory specification, invocation policy, or checkers, and a validation gate decides which updates enter subsequent execution (Yu et al., 2026). Its repair target is a memory-control component. `EpiGraph` makes the execution dependency the repair target: a contradicted expectation identifies the graph descendants whose prerequisites or outputs must be revised. The biomedical cases in Appendix E.5 make these dependencies visible through label checks and statistical receipts.

Graph Engineering. Graph engineering makes relationships among agent components, task structure, and execution state an explicit object of design. Its research questions include how to search and transform reasoning states, how to coordinate models and tools, how to schedule and route work, and how to use structured memory for later decisions (Yao et al., 2023a; Besta et al., 2023; Yun et al., 2026; Yu et al., 2025; Feng et al., 2026a). These approaches respond to the increasing length and heterogeneity of agent workflows: thoughts must be refined, messages must be exchanged, specialized components must be assigned, and state must persist across steps. Surveys place these mechanisms within a broader design space spanning planning, execution, memory, coordination, verification, provenance, and recovery (Bei et al., 2025; Feng et al., 2026b).

The main solution strategies differ in what the graph controls. Reasoning graphs transform intermediate states; agent-composition graphs organize communication and delegation; workflow graphs expose task structure and scheduling; and graph memories or routers select models, roles, or subsequent actions (Schick et al., 2023; Yun et al., 2026; Wang et al., 2026b; Yu et al., 2025; Feng et al., 2026a). Harness-oriented systems additionally make tools, interfaces, state transfer, and validation programmable (Yang et al., 2024; Pan et al., 2026; Wang et al., 2026a; Zhang et al., 2026). Within this landscape, `EpiGraph` makes the conditions for valid execution the engineering object. The central question is what must be established before an action may proceed and what observed evidence permits its result to be accepted. Prerequisites, expected observations, and receipts are therefore first-class nodes: `Guards` makes a prerequisite binding on action admission, `Predicts` declares the observable commitment of that action, and `Verifies` connects a receipt to the commitment it discharges. This gives dependency structure an operational account of legality: an edge specifies the condition that licenses a transition, beyond its position in a reasoning or communication sequence. The term meta-graph denotes this layer of execution dependencies. Its six operators—`Compile`, `Expand`, `Prune`, `Admit`, `Close`, and `Revise`—retain their semantics across domain contents, models, and harnesses, allowing execution obligations to be engineered independently of the system that carries out an action. A cohort-membership check and a repository-state check instantiate different predicates but the same admission operation; a contradicted receipt identifies which dependent branch requires revision while preserving valid upstream work. Scientific reasoning gives this abstraction a demanding, concrete setting: statistical tests, multiple-testing corrections, label-consistency checks, and saved numerical results provide machine-checkable acceptance criteria. For example, feature ranking and corrected significance discharge distinct obligations in the metabolomics analysis, while time-label checks determine whether the proteomics analysis needs a sensitivity branch (Appendix E.5). These mechanisms connect the three claims at a common level: legality resides in the `Guards`, `Predicts`, and `Verifies` edges, portable operators engineer the graph above individual tasks, and scientific receipts determine whether declared obligations have been met.

B FORMALIZATION DEFINITION OF EPIGRAPH

`EpiGraph` engineers an execution meta-graph using six operators: `Compile`, `Expand`, `Prune`, `Admit`, `Close`, and `Revise`. The graph is meta-level because its nodes describe the conditions of valid task execution and carry the domain-specific contents needed to evaluate them. The operators use the host agent's native tools, state, and lifecycle. At inference time, `EpiGraph` supplies the host agent with an execution policy that encodes these operators and the named graph-edge dependencies. The policy guides how the agent checks prerequisites, anticipates observations, collects evidence, and revises subsequent actions. Algorithm 1 formalizes this graph-based execution logic.

B.1 EXECUTION PRIMITIVES

We first separate actions, observations, and persisted evidence. This distinction supplies the vocabulary for the execution contract and prevents a proposed outcome from being treated as an observed result. An agent receives a task specification x and observes an initial task state s_0. At step t, it selects action a_t, receives observation o_t, and may persist evidence e_t, such as a test result, output file, diff, or evaluator receipt. Its observable execution trace is

$$\tau_T = \big((a_1, o_1, e_1), \ldots, (a_T, o_T, e_T)\big), \tag{5}$$

where $e_t = \varnothing$ when an action produces no durable evidence. We divide actions into information actions $\mathcal{A}^I$, which reduce uncertainty without intentionally changing the target state, and world actions $\mathcal{A}^W$, which edit, execute, dispatch, or otherwise change it.

The evaluator maps the terminal artifact and accumulated evidence $\mathcal{E}_T = \{e_t : 1 \le t \le T,\ e_t \neq \varnothing\}$ to a task-native score y. `EpiGraph` exposes the dependency between task intent, actions, and $\mathcal{E}_T$ before the evaluator is invoked.

From an agent's next action to execution dependencies. Suppose the agent is asked to compare measurements before and after treatment. Before running a paired test, it needs to establish which measurements belong to the same subject. It must also specify what the test should produce and which saved output will support its conclusion. These are three different objects: a `Prerequisite` p_t (matched subjects), an `Expected observation` $\hat{o}_t$ (a test result over the matched cohort), and a `Receipt` e_t (the actual saved result). The receipt is unavailable when the action is planned; the required receipt $e_t^\star$ can already be specified.

Why represent these dependencies as a graph? The subject-matching check can guard several downstream analyses. A mismatch in subject identity invalidates those analyses while leaving unrelated data checks intact. A graph makes this shared dependency and the scope of repair explicit. In our notation, $p_t \xrightarrow{\texttt{Guards}} a_t$ licenses the action, $a_t \xrightarrow{\texttt{Predicts}} \hat{o}_t$ records its expected observation, and $e_t \xrightarrow{\texttt{Verifies}} \hat{o}_t$ records that the inspected receipt satisfies the declared acceptance predicate. Before execution, this relation specifies the evidence to obtain and check.

Definition: execution meta-graph. Let $C = (G, B, R)$ be a task contract: G specifies the goal, B the execution boundaries, and R the required evidence. An execution meta-graph is a typed directed graph $\mathcal{G}_t = (V_t, E_t)$ whose four execution node classes are prerequisites P_t, actions A_t, expected observations O_t, and receipts R_t. The three obligation classes P_t, O_t, and R_t carry the conditions and evidence of the task; A_t contains the transition vertices that are admitted between them. Its named edge types connect these nodes while the graph is maintained under C:

$$V_t = P_t \,\dot\cup\, A_t \,\dot\cup\, O_t \,\dot\cup\, R_t \,\dot\cup\, U_t \,\dot\cup\, D_t, \qquad E_t \subseteq V_t \times \mathcal{K}_{\mathrm{EpiGraph}} \times V_t, \tag{6}$$

where U_t and D_t are unresolved-uncertainty and decision annotations and $\mathcal{K}_{\mathrm{EpiGraph}}$ is the set of graph-edge labels. A task contract instantiates one graph; changing the task's deliverables, prerequisites, or acceptance evidence produces a different graph instance. The collection of task-level graph instances is the execution structure of agentic scientific reasoning, with scientific receipts providing the machine-checkable observations that connect each task's execution to its conclusion. The prefix meta identifies the agent's control level: the graph represents the conditions for using the data, tools, and results of a task. For example, patients and measurements are domain contents; permission to run a paired analysis and evidence for accepting its result are execution dependencies. In this sense, a meta-graph is a control graph over task execution with explicit node classes and named edge types: its vertices carry obligations and actions, and its edges state when a transition may be taken

or its result accepted. The visual vocabulary follows the same typing. Blue sans-serif nodes denote `Prerequisite` conditions, ochre nodes denote `Expected observation` commitments, and plum nodes denote `Receipt` evidence; each node type uses a matching pale fill and border. Edge labels use the same type colors for `Guards`, `Predicts`, and `Verifies`. Operator labels use a six-color palette: blue `Compile`, ochre `Expand`, slate `Prune`, teal `Admit`, plum `Close`, and red `Revise`. This vocabulary is used in the overview, notation, algorithm, and every case-study graph. This separation relates to precondition–postcondition reasoning (Hoare, 1969). Section 3 develops six operators for maintaining these dependencies. Table 2 maps the symbols to their definitions and the implementation's node types.

B.2 Execution Edge Types

Figure 1 gives the visual summary before the formal definitions. A task contract supplies a goal, execution boundary, and evidence predicate.

The execution meta-graph makes the dependencies that govern each action explicit. Prerequisites, expected observations, and receipts are first-class nodes; the named edge types state when an action is licensed, what it commits to produce, and which persisted evidence closes that commitment.

`Guards`. A prerequisite-to-action edge $p_t \xrightarrow{\texttt{Guards}} a_t$ licenses an action after its support conditions are satisfied. A support set may contain several conjunctive requirements, while alternative support sets provide distinct legal paths.

`Predicts`. An action-to-expectation edge $a_t \xrightarrow{\texttt{Predicts}} \hat{o}_t$ records the quantities, output schema, and decision rule expected from the action before it runs.

`Verifies`. An evidence-to-expectation edge $e_t \xrightarrow{\texttt{Verifies}} \hat{o}_t$ records that an inspected receipt satisfies the declared acceptance predicate. The receipt is produced by execution; its required form can be declared in advance.

Graph state. Let $C = (G, B, R)$ be the task contract. The execution graph is

$$V_t = P_t \dot{\cup} A_t \dot{\cup} O_t \dot{\cup} R_t \dot{\cup} U_t \dot{\cup} D_t, \qquad E_t \subseteq V_t \times \mathcal{K}_{\text{EpiGraph}} \times V_t, \tag{7}$$

where

$$\mathcal{K}_{\text{EpiGraph}} := \{\texttt{REQUIRES}, \texttt{INFORMS}, \texttt{ENABLES}, \texttt{Guards}, \texttt{Predicts}, \texttt{Verifies}, \texttt{REVISES}\}$$

is the set of graph-edge labels, distinct from $\mathcal{O}_{\text{EpiGraph}}$, the set of six graph operators. Here P_t, A_t, O_t, R_t are prerequisite, action, expected-observation, and receipt nodes, and U_t, D_t annotate unresolved uncertainty and decisions. The visual vocabulary uses matching colors for node classes and for `Guards`, `Predicts`, and `Verifies` labels. Table 2 fixes the symbols used by the policy; the full action–observation–evidence setup appears in this appendix.

B.3 Six Operators

The operators are the graph-construction semantics. `Compile` creates the contract, prerequisite nodes, expected observations, and receipt obligations; `Expand` adds candidate paths and their alternatives; `Prune` sequentially removes redundant candidates while preserving compliant outcomes; `Admit` marks an action available only when its named edge conditions are satisfied; `Close` adds the observed result and its verifying receipt; and `Revise` rewrites the descendants of a contradicted node while retaining independently supported branches. Thus the graph instance is determined by the task contract and its evidence obligations, while the six operator meanings remain fixed. At any execution state the instance is a branching dependency DAG. The observation argument in each transition is the newly returned state: it can satisfy an expected observation and enable closure, or contradict one and select the `Revise` operator. We formalize execution as a sequence of operator-labelled graph-state transitions:

$$\begin{aligned} \mathcal{G}_{t+1} &= o_t(\mathcal{G}_t, s_t, o_t^{\text{obs}}, e_t), \\ o_t &\in \mathcal{O}_{\text{EpiGraph}} = \{\texttt{Compile}, \texttt{Expand}, \texttt{Prune}, \texttt{Admit}, \texttt{Close}, \texttt{Revise}\}. \end{aligned} \tag{8}$$

The operator selected at a state follows the available evidence and unresolved obligations. An observation can close a branch, while a contradiction revises the affected descendants and preserves

any branch with independent support. Prerequisites, expected observations, and receipts are first-class nodes. An action is legal when at least one satisfied support set reaches it through its `Guards` edges, it reaches an expected observation through a `Predicts` edge, and a declared receipt can close its effect through a `Verifies` edge. The six operators transform decision dependencies at the meta-graph level. Their semantics are shared across biomedical, software, and other task domains; the host harness and model instantiate node contents and execute admitted actions. Scientific receipts such as cohort counts, corrected statistical tests, confidence intervals, model diagnostics, and persisted artifacts support explicit acceptance predicates. Code can check these predicates against recorded evidence: for example, whether the paired cohort has the declared size or a corrected test meets the stated threshold. Scientific interpretation remains tied to the chosen analysis and its assumptions. Appendix H reproduces the executable policy used to instantiate these principles and operators.

`Compile`: To make the execution contract explicit, `Compile` maps task x and observed state s_0 to

$$C(x, s_0) = \big(G, B, R\big), \tag{9}$$

where G is a predicate over the requested terminal state and deliverables, B specifies admissible data, paths, interfaces, authority, side effects, resources, invariants, and non-goals, and R is a predicate over the evidence required by the benchmark or user. A field is not filled with a convenient assumption: if its value could alter the next decision, it becomes an explicit uncertainty $u \in \mathcal{U}_t$ and must be resolved by a bounded information action before the dependent world action.

Compilation defines success as a conjunction:

$$\mathrm{Complete}_C(\tau_T, s_T, \mathcal{E}_T) = G(s_T) \ \wedge \ B(\tau_T) \ \wedge \ R(\mathcal{E}_T). \tag{10}$$

Consequently, a plausible response, a zero exit code, or a worker's declaration of completion cannot substitute for a missing receipt.

The contract seeds a small mutable graph

$$\begin{aligned} &\mathcal{G}_t = (V_t, E_t), \\ &V_t = P_t \,\dot\cup\, A_t \,\dot\cup\, O_t \,\dot\cup\, R_t \,\dot\cup\, U_t \,\dot\cup\, D_t, \\ &E_t \subseteq V_t \times \mathcal{K}_{\mathrm{EpiGraph}} \times V_t. \end{aligned} \tag{11}$$

The four core node classes carry the task's execution contract: P_t contains requirements and observed states used as guards, A_t contains information and world actions, O_t contains the expected observations declared before an action, and R_t contains the persisted receipts that can discharge them. U_t and D_t record unresolved uncertainties and decisions that steer the graph. The decisive claim is structural: prerequisites, expected observations, and receipts are first-class nodes, and action legality is evaluated from the `Guards`, `Predicts`, and `Verifies` edge predicates. The graph is represented in the host task state and its action–observation–evidence events with these node and edge semantics.

`Expand`: To expose alternatives with decision value, `Expand` composes four reasoning postures. Backward expansion starts at G and R and asks which prerequisites and actions can produce them. Forward expansion starts at s_t and enumerates reachable actions and observations. Dialectical expansion compares alternative interpretations and identifies an observation that would distinguish them. Expectation expansion attaches a predicted observation and a failure implication to a candidate action. Their composition produces candidate paths.

`Prune`: This operator removes redundant candidate structure one node at a time while preserving compliant terminal outcomes. Let $\mathrm{Out}_C(H)$ denote the outcomes induced by continuation paths satisfying the fixed contract C in the current working graph H. The relevance test is

$$\mathrm{Rel}(v; H) = \mathbb{I}[\mathrm{Out}_C(H) \neq \mathrm{Out}_C(H \setminus v)]\,. \tag{12}$$

Starting with $H_t^{(0)} = \mathcal{G}_t$, a pruning step is

$$H_t^{(j+1)} = H_t^{(j)} \setminus v_j, \quad \mathrm{Rel}(v_j; H_t^{(j)}) = 0, \quad \mathrm{Out}_C(H_t^{(j)}) \neq \varnothing. \tag{13}$$

Each step recomputes relevance on the updated graph. Thus $\mathrm{Out}_C(H_t^{(j)}) = \mathrm{Out}_C(\mathcal{G}_t)$ throughout the pass, preserving at least one continuation to each reachable compliant outcome. Candidate order

follows the host's cost and risk priorities. The pass retains contract obligations and persisted receipts, and returns its final working graph to admission. When compliant outcomes are still unresolved, expansion and information actions establish the next candidates.

`Admit`: This operator keeps information actions $\mathcal{A}^I$ distinct from world actions $\mathcal{A}^W$. For each non-trivial candidate a, `EpiGraph` records the requirement it supports, alternatives, uncertainty reduction, expected observation, evidence, cost, and risk. The omission test marks a necessary relative to the current contract when removing it eliminates every compliant completion path:

$$\mathrm{Nec}_t(a) = \mathbb{I}[\Pi_C(\mathcal{G}_t) \neq \varnothing \ \wedge \ \Pi_C(\mathcal{G}_t \setminus a) = \varnothing]\,. \tag{14}$$

A diagnostic action is admitted when its possible observations select different continuation paths. Let $\mathrm{Viable}_t(a)$ indicate that at least one compliant continuation begins with a. Necessary and diagnostic actions receive priority; when no such action is available, a viable optional alternative remains admissible and the host chooses among those alternatives by cost, risk, reversibility, or its native policy. Optional denotes an action that becomes unnecessary once an equivalent alternative has supplied the same contract or evidence. Redundant actions are pruned when an already completed alternative supplies equivalent evidence. Legality and necessity are separate predicates. Define

$$\mathrm{Legal}_t(a) = B(\tau_t \oplus a) = 1 \ \wedge \ \mathrm{EdgeReady}_t(a) \ \wedge \ \mathrm{Viable}_t(a). \tag{15}$$

The edge-gated candidate set is

$$\mathcal{A}_t^{\mathrm{adm}} = \{a \in \mathcal{A}(s_t) : \mathrm{Legal}_t(a)\}\,. \tag{16}$$

The host first considers the legal subset satisfying $\mathrm{Nec}_t(a) \vee \mathrm{Diag}_t(a)$ when that subset is nonempty; otherwise it chooses from the full legal set. Necessity and diagnostic value therefore order legal actions; they do not remove a legal action merely because an alternative compliant continuation exists. Here $\mathrm{EdgeReady}_t(a)$ requires that at least one incoming `Guards` support set is satisfied, a `Predicts` edge declares the expected observation, and a `Verifies` obligation specifies the receipt required for closure. Admission establishes this obligation before execution; closure discharges it using observed evidence. Edge predicates are checked as part of admission and closure. The host remains free to choose among legal actions by cost, risk, reversibility, or its native policy.

Support can be conjunctive or alternative. For a node v, let $\mathcal{S}_H(v)$ be the family of support sets encoded by the incoming relations in graph H. For a Boolean validity assignment σ, define

$$\mathrm{Support}(v; H, \sigma) = \bigvee_{S \in \mathcal{S}_H(v)} \bigwedge_{u \in S} \sigma(u). \tag{17}$$

Different sets S represent alternative supports; members of one set are joint obligations. Each S identifies a complete support combination. Removing a required member disables that combination; each surviving combination retains all its original requirements. An empty support family is false; an explicitly unconditional node has the support family $\{\varnothing\}$. The assignment $\sigma_t(u) = 1$ means that u is currently established and valid; unresolved or invalid nodes have value zero. For admission, this predicate applies to the incoming `Guards` support sets. `Revise` evaluates the same predicate with updated validity assignments.

Before execution, admission also produces a bounded local contract

$$L_t = \big(q_t, b_t, m_t, \hat{o}_t, e_t^\star, \rho_t\big), \tag{18}$$

where q_t is the global requirement supported by the action, $b_t \subseteq B$ is its local boundary, m_t is its pass metric, $\hat{o}_t$ is the expected observation, $e_t^\star$ is the required receipt, and ρ_t specifies retry, replan, escalate, or stop behavior on mismatch. The same contract is passed to a delegated worker. A worker may return observations and evidence, but cannot relax B, redefine R, or declare global completion.

`Close`: This operator compares observation o_t and persisted receipt e_t against L_t. Here $\hat{o}_t$ specifies the quantities to estimate, output schema, and decision rule to evaluate. Let $\mathrm{Accept}_t(o_t, e_t; L_t)$ denote that the observation has the declared schema, the local metric m_t is evaluable and passes, and the receipt supplies the declared evidence. A branch closes exactly when

$$\mathrm{Accept}_t(o_t, e_t; L_t) = 1. \tag{19}$$

At task level, verified completion additionally requires Equation 10. A missing statistical test, output artifact, or evaluator receipt therefore produces a partial, blocked, or unverified state. For a yes/no

scientific question, let $\phi_t(o_t) \in \{0, 1\}$ be the declared decision rule and z_t the reported verdict. Closure checks $z_t = \phi_t(o_t)$ together with the estimation, schema, and evidence requirements in L_t. Both verdicts are valid outputs. A null, non-significant, or empty discovery result closes the branch when these execution requirements are satisfied.

Scientific tasks instantiate three kinds of evidence obligation. Artifact checks establish schema, completeness, and agreement between saved results and reported summaries. Computational checks compare a calculation with a specified reference or recomputation, recording the inputs, procedure, and comparison criterion. Assumption checks and sensitivity analyses examine cohort definitions, analysis units, and model conditions; scientific interpretations are stated under these conditions. Each `Verifies` edge binds a receipt to the particular predicate in L_t that it satisfies. Code or an evaluator checks that predicate against the recorded evidence. A receipt closes the declared evidence obligation and establishes internal consistency of the saved artifact; its closure is the evidence relation used by the policy.

`Revise`: This operator identifies the unmet execution obligation, such as an invalid cohort join, failed fit, or missing evidence, and preserves its receipt. It propagates loss of support from the affected node h_t. Descendants are taken along support dependencies in the current graph; `Revise` arrows denote transitions between graph states. On the current finite dependency DAG, set $\mathcal{I}_t^{(0)} = \{h_t\}$ and define

$$\sigma_t^{(j)}(u) = \sigma_t(u)\, \mathbb{I}[u \notin \mathcal{I}_t^{(j)}], \tag{20}$$

$$\mathcal{I}_t^{(j+1)} = \mathcal{I}_t^{(j)} \cup \left\{ v \in \mathrm{Desc}_{\mathcal{G}_t}(h_t) : \begin{array}{l} \sigma_t(v) = 1, \\ \mathrm{Support}(v; \mathcal{G}_t, \sigma_t^{(j)}) = 0 \end{array} \right\}. \tag{21}$$

The increasing invalidation sets stabilize at $\mathcal{I}_t^\star$ after at most $|V_t|$ strict additions. Previously unresolved nodes retain their unresolved status. An established descendant remains valid while at least one complete support set remains satisfied. The `Revise` step is

$$\mathcal{G}_{t+1} = \mathrm{Update}(\mathrm{Invalidate}(\mathcal{G}_t, \mathcal{I}_t^\star), o_t, e_t), \quad \mathrm{Accept}_t(o_t, e_t; L_t) = 0. \tag{22}$$

Invalidate marks the affected nodes invalid while retaining their records and receipts. Update incorporates the new observation and receipt and exposes revised candidate dependencies; repaired branches regain validity through admission and closure. The agent then chooses an information action, retry, alternative branch, or explicit stop state. Together, the six operators make the graph independently engineerable above task and model: domains instantiate node content and receipts, whereas the compile–expand–prune–admit–close– revise semantics remain unchanged. All reported nodes are grounded in observable actions and receipts.

B.4 Notation and execution policy

C Experimental Environment and Protocol

This appendix first fixes the experimental environment and registered comparison settings. The following sections group evidence by research question, keeping outcome measurements, process evidence, module effects, and prospective interventions separate. Each comparison retains its task identities and native scoring rule.

C.0.1 Dataset and Benchmark Descriptions

- **GPQA-Diamond**(Rein et al., 2024)[1] is a graduate-level, multiple-choice science benchmark designed around questions that are difficult to answer by ordinary web search and require substantial knowledge in biology, physics, and chemistry. The split used here contains 198 questions, and we report exact-answer accuracy.
- **AIME 2024**[2] and **AIME 2025**[3] are annual competition-math collections whose answers are integers in the AIME answer range. Each set used here contains 30 problems, and we report exact-match accuracy.

[1] https://huggingface.co/datasets/Idavidrein/gpqa
[2] https://huggingface.co/datasets/HuggingFaceH4/aime_2024
[3] https://huggingface.co/datasets/opencompass/AIME2025

Table 2: Notation used by `EpiGraph`, grouped by execution, graph structure, and operator semantics. Symbols agree with this appendix, Equations 12–22, and Algorithm 1.

Symbol	Meaning
$x; t, T$	Task specification; action index and terminal index.
$s_t; a_t, o_t, e_t$	Observed state; action, resulting observation, and persisted receipt. $e_t = \varnothing$ denotes no receipt.
$\tau_t; \mathcal{E}_t; y$	Trace through step t; set of nonempty receipts; evaluator-native score.
$\mathcal{A}(s_t), \mathcal{A}^I, \mathcal{A}^W$	Available actions; information actions; world actions.
$C = (G, B, R)$	Contract: goal predicate, boundary constraints, and evidence predicate; $\mathcal{G}_t$ is the graph instantiated by this contract.
Complete_C	Conjunction of goal attainment, boundary compliance, and evidence satisfaction (Eq. 10).
$\mathcal{G}_t = (V_t, E_t)$	Execution meta-graph with explicit node classes and labeled directed edges.
P_t, A_t, O_t, R_t	Execution node classes: prerequisites, actions, expected observations, and receipts; P_t, O_t, R_t are obligation classes.
U_t, D_t	Unresolved-uncertainty and decision annotations that steer graph maintenance.
$p_t, \hat{o}_t, e_t^\star$	Prerequisite, expected observation, and required receipt specification. A star denotes a requirement, not an observed receipt.
`Guards`	Prerequisite-to-action relation checked at admission.
`Predicts`	Action-to-expectation relation specifying quantities, output schema, and the decision rule before execution.
`Verifies`	Evidence-to-expectation relation established when a receipt satisfies its declared acceptance predicate.
$\mathcal{K}_{\text{EpiGraph}}$	Set of named graph-edge labels in the execution meta-graph.
`REQUIRES`, `INFORMS`, `ENABLES`, `REVISES`	Supporting dependency, information, enabling, and revision relations from the policy.
$\Pi_C(\mathcal{G}_t); \text{Out}_C(\mathcal{G}_t); v$	Compliant continuation paths, their terminal outcomes, and a candidate node.
$\text{Rel}(v; H), \text{Nec}_t(a), \text{Legal}_t(a)$	Relevance, necessity, and legality predicates (Eqs. 12, 14, 15).
$\text{Support}(v; H, \sigma)$	Support predicate under validity assignment σ, with conjunction within a support set and alternatives across sets (Eq. 17).
$\text{Viable}_t(a)$	True when at least one compliant continuation begins with action a.
$\text{Diag}_t(a)$	True when possible observations distinguish continuation paths.
$\text{EdgeReady}_t(a)$	At least one `Guards` support set satisfied, with expectation declared and required verification receipt specified.
$\mathcal{A}_t^{\text{adm}}$	Legal actions satisfying boundary, edge, and viability conditions (Eq. 16).
$\phi_t(o_t); z_t$	Declared scientific decision rule and reported yes/no verdict; closure checks their agreement.
$\text{Accept}_t(o_t, e_t; L_t)$	Local closure predicate: declared observation schema, passing metric, and required receipt (Eq. 19).
$L_t = (q_t, b_t, m_t, \hat{o}_t, e_t^\star, \rho_t)$	Local contract: supported requirement, local boundary, pass metric, expectation, required receipt, and mismatch response.
$h_t; \mathcal{I}_t^{(j)}; \mathcal{I}_t^\star$	Contradicted node; invalidation set at propagation step j; stabilized invalidation set (Eq. 21).
$H_t^{(j)}; \sigma_t^{(j)}$	Working graph after j sequential pruning steps; validity assignment after j invalidation steps.
$\mathcal{S}_H(v)$; Desc	Complete alternative support sets for v; descendants along dependency relations.
Invalidate; Update	Mark affected nodes invalid while retaining records; incorporate observations, receipts, and revised candidates.
$\oplus, \setminus, \models, \mathbb{I}$	Trace extension; removal; satisfaction of a declared condition; binary indicator.

- **BeyondAIME**[4] is a manually revised collection of 100 competition problems that extends the difficulty and diversity of AIME-style evaluation. We use its integer-answer format and report exact-match accuracy.

[4] https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME

Algorithm 1 The six-operator `EpiGraph` execution policy

Require: task x, initial state s_0, finite execution budget β
Ensure: verified artifact and receipts, or an explicit unresolved state

`Compile` $C = (G, B, R)$; seed first-class prerequisite, expectation, and receipt nodes; set $\beta_0 = \beta$
while $\beta_t > 0$ and $\neg\,\mathrm{Complete}_C(s_t, \tau_t, \mathcal{E}_t)$ **do**
  `Expand` candidate dependencies backward, forward, dialectically, or by expectation
  `Prune` candidates sequentially using Eq. 13; recompute relevance after each deletion
  if a decisive prerequisite is unknown **then**
    Add the smallest bounded information action that can resolve it
  end if
  if $B(\tau_t) = 0$ **then**
    Stop further side effects and return the boundary violation with its receipt
  end if
  `Admit` edge-ready actions on at least one compliant continuation; prioritize necessary or diagnostic actions when available
  if $\mathcal{A}_t^{\mathrm{adm}} = \varnothing$ **then**
    Return blocked with the unresolved contract and verified evidence
  end if
  Select candidate action a_t, expected observation $\hat{o}_t$, and required receipt $e_t^\star$
  Execute a_t through the host interface; persist observation o_t and receipt e_t
  Update β_{t+1} by the action's declared cost
  if $B(\tau_t \oplus a_t) = 0$ **then**
    Stop further side effects and return the boundary violation with its receipt
  else if $\mathrm{Accept}_t(o_t, e_t; L_t) = 1$ **then**
    `Close`: attach $e_t \xrightarrow{\texttt{Verifies}} \hat{o}_t$, record the result and its verdict, close the branch, and update satisfied nodes
  else
    `Revise`: identify the unmet execution obligation; propagate support loss to $\mathcal{I}_t^\star$ (Eq. 21); mark invalid nodes and update dependencies (Eq. 22)
  end if
end while
if $\mathrm{Complete}_C(s_t, \tau_t, \mathcal{E}_t)$ **then**
  return verified completion with receipts
else
  return partial or blocked with the unresolved contract and verified evidence
end if

- **HMMT 2025 and HMMT 2026**[5] are contest-mathematics sets containing 30 and 33 tasks, respectively. They provide additional problems with exact numerical answers.
- **BiomniBench-DA**(Qu et al., 2026)[6] is a public 50-task data-analysis subset for complete biomedical research workflows. Each task supplies a dataset, a research question, and an expert-authored rubric that scores data handling, analysis decisions, quantitative results, statistical evidence, biological interpretation, and source-grounded reporting. Gemini 3.1 Pro is used as the rubric judge, and partial completion receives partial task credit.
- **CMT-Bench**(Pan et al., 2025)[7] contains 50 expert-authored condensed-matter theory problems spanning Hartree–Fock, quantum Monte Carlo, and tensor-network calculations. We report exact task accuracy and the category breakdown supplied by the benchmark.
- **SWE-bench Verified Mini**(Jimenez et al., 2024)[8] contains 50 real GitHub issues, with 25 Django and 25 Sphinx tasks. A task is correct when the submitted patch satisfies the repository's tests and the benchmark evaluator accepts the resulting repository state.

[5] https://www.hmmt.org/www/archive/
[6] https://huggingface.co/datasets/phylobio/BiomniBench-DA
[7] https://arxiv.org/abs/2510.05228
[8] https://www.swebench.com/

The registered reasoning effort is xhigh for CMT-Bench and BiomniBench-DA and medium for all other benchmark campaigns.

C.0.2 Agent Systems Descriptions

- **Codex**[9] is OpenAI's local coding agent. The CLI runs on the user's computer, reads and edits repositories, executes commands, and supports software-engineering workflows; we evaluate GPT 5.4, GPT 5.5, GPT 5.6 Terra, and GPT 5.6 Sol.
- **Claude Code**[10] is Anthropic's terminal-based coding agent. It inspects a codebase, edits files, runs development commands, and supports multi-step software tasks; our campaigns use Claude Haiku 4.5, Claude Sonnet 4.6, Claude Sonnet 5, and Claude Opus 5.
- **Pi**[11] is an open-source, extensible coding-agent harness. Its repository provides an interactive coding-agent CLI, a tool-calling agent core with state management, and a unified multi-provider model interface; we use it as a lightweight independent host for testing transfer across harnesses.
- **Biomni**(Huang et al., 2025)[12] is a biomedical AI agent that combines domain tools, datasets, and analysis resources for end-to-end biomedical research workflows.
- **EvoScientist**[13] is a self-evolving AI scientist designed for autonomous research. Its workflow combines specialized sub-agents for planning, research, coding, debugging, analysis, and writing, together with persistent memory, tool use, and iterative evaluation; it appears in the released BiomniBench-DA comparison.
- **OpenScience (Synthetic Sciences)**[14] is an open-source AI workbench for scientific research. It lets a research agent search literature, write and run code, execute experiments, and preserve the resulting files and trace for review.
- **OpenScience (AI4S Research)**(The Open Science Desktop Contributors, 2026)[15] is the AI4S Research project now named Open Science Desktop. It provides a local-first research workbench for organizing scientific tasks, evidence, and reproducible project artifacts.
- **Wisp Science**(Xu et al., 2026b)[16] is an open-source, local-first AI research workbench. It combines literature search, Python and R execution, scientific database access, evidence capsules, and persistent project artifacts.

D RQ1: Outcome-Level Evaluation

The outcome-level supplement expands the main benchmark results before turning to process-level evidence. It covers benchmark breakdowns, coding-agent transfer, and resource measurements under the registered settings.

D.1 Outcome-Level Benchmark Breakdown

We retain the registered model, reasoning effort, and task set for each comparison. CMT-Bench is grouped by condensed-matter methods; GPQA-Diamond by scientific subject; and AIME by source examination. These divisions expose different kinds of reasoning demand. The paragraph accompanying each table identifies the principal pattern and relates it to the corresponding overall result. **CMT-Bench: Overall and Task Categories.** Table 3 compares Pi, Codex, and Claude Code under the baseline and + `EpiGraph` conditions over the same 50-task benchmark, with overall and eight task-category groups arranged side by side. Pi uses max effort, Codex uses xhigh, and Claude Code uses max. Overall accuracy uses all 50 tasks.

[9] https://github.com/openai/codex
[10] https://code.claude.com/docs/en/overview
[11] https://github.com/badlogic/pi-mono
[12] https://github.com/biomni-dev/biomni
[13] https://github.com/EvoScientist/EvoScientist
[14] https://github.com/synthetic-sciences/openscience
[15] https://github.com/ai4s-research/open-science
[16] https://github.com/xuzhougeng/wisp-science

Table 3: CMT-Bench accuracy (%) overall and across all eight task categories, comparing the baseline and + `EpiGraph` settings. Changes are in pp. Green upward arrows indicate improvement; red downward arrows indicate deterioration. Pi and Claude Code use max effort; Codex uses xhigh.

	Methods	Setting	Overall (50)	HF (5)	VMC (2)	DMRG (4)	QMC (6)	ED (8)	PEPS (3)	SM (6)	Other (16)
Pi	GPT 5.4	-	22.00	0.00	0.00	0.00	16.67	37.50	33.33	50.00	18.75
		+EpiGraph	**36.00**	**20.00**	**0.00**	**0.00**	**50.00**	**62.50**	**33.33**	**33.33**	**37.50**
		Δ	↑ 14.00 pp	↑ 20.00 pp	0.00 pp	0.00 pp	↑ 33.33 pp	↑ 25.00 pp	0.00 pp	↓ 16.67 pp	↑ 18.75 pp
	GPT 5.5	-	30.00	0.00	0.00	25.00	16.67	50.00	33.33	50.00	31.25
		+EpiGraph	**42.00**	**20.00**	**0.00**	**25.00**	**50.00**	**62.50**	**66.67**	**33.33**	**43.75**
		Δ	↑ 12.00 pp	↑ 20.00 pp	0.00 pp	0.00 pp	↑ 33.33 pp	↑ 12.50 pp	↑ 33.33 pp	↓ 16.67 pp	↑ 12.50 pp
	Claude Sonnet 4.6	-	10.00	20.00	0.00	0.00	0.00	12.50	0.00	33.33	6.25
		+EpiGraph	**14.00**	**0.00**	**50.00**	**0.00**	**0.00**	**37.50**	**0.00**	**33.33**	**6.25**
		Δ	↑ 4.00 pp	↓ 20.00 pp	↑ 50.00 pp	0.00 pp	0.00 pp	↑ 25.00 pp	0.00 pp	0.00 pp	0.00 pp
Codex	GPT 5.5	-	34.00	40.00	0.00	50.00	16.67	75.00	33.33	33.33	18.75
		+EpiGraph	**36.00**	**40.00**	**0.00**	**0.00**	**33.33**	**75.00**	**66.67**	**33.33**	**25.00**
		Δ	↑ 2.00 pp	0.00 pp	0.00 pp	↓ 50.00 pp	↑ 16.67 pp	0.00 pp	↑ 33.33 pp	0.00 pp	↑ 6.25 pp
Claude Code	Claude Sonnet 4.6	-	32.00	20.00	0.00	0.00	0.00	37.50	66.67	50.00	43.75
		+EpiGraph	**36.00**	**20.00**	**50.00**	**25.00**	**0.00**	**75.00**	**33.33**	**50.00**	**31.25**
		Δ	↑ 4.00 pp	0.00 pp	↑ 50.00 pp	↑ 25.00 pp	0.00 pp	↑ 37.50 pp	↓ 33.33 pp	0.00 pp	↓ 12.50 pp

GPQA-Diamond: Scientific Domains. Table 4 shows where the aggregate gains arise. For Claude Sonnet 5, chemistry improves by 35.48 pp, compared with 3.49 pp in physics and 15.79 pp in biology. Claude Opus 5 makes its largest gain in biology (36.84 pp). GPT 5.5 improves in chemistry and physics but declines by 5.26 pp in biology. Latency also varies: Claude Sonnet 5 becomes faster in chemistry and physics but takes longer in biology. **AIME 2024: Examination-Level Breakdown.** Table 5 separates the two source examinations, each with 15 tasks. GPT 5.5 and GPT 5.6 Sol reach 100% on both examinations with `EpiGraph`. For Claude Opus 5, the larger accuracy gain occurs on AIME II: the baseline solves 10 of 15 tasks, while `EpiGraph` solves 14, a gain of 26.67 pp. Claude Haiku 4.5 also concentrates its aggregate improvement on AIME II, rising from 80.00% to 93.33% while mean time falls from 94.02 to 70.06 seconds. The split separates accuracy gains from latency changes: GPT 5.6 Terra retains the same accuracy on both examinations, with faster execution on AIME I and slower execution on AIME II.

AIME 2025: Examination-Level Breakdown. In Table 6, all three Codex configurations reach 100% on both examinations with `EpiGraph`. GPT 5.6 Terra improves from 73.33% to 100% on AIME I and from 66.67% to 100% on AIME II, while mean times fall from 86.80 to 49.83 seconds and from 107.54 to 43.54 seconds, respectively. Claude Opus 5 improves accuracy and reduces time on both examinations; its AIME I mean time falls from 154.84 to 61.59 seconds. The latency pattern is model-specific: GPT 5.5 and GPT 5.6 Sol solve every AIME II task with `EpiGraph` but take longer on that examination than their baselines.

D.2 Coding-Agent Transfer

The appendix evaluation contains 50 repository issues: 25 Django and 25 Sphinx tasks, all retained in the denominator (Jimenez et al., 2024). We compare Codex with GPT 5.5 at medium effort and Claude Code with Claude Sonnet 5 at medium effort. Figure 8 visualizes the outcomes and paired decomposition. Normalized gain divides the accuracy increase by the baseline remaining error fraction, following the normalized-gain construction (Hake, 1998).

D.3 RQ1 Supplement: Outcome-Level Efficiency Measurements

Execution time and token consumption are recorded for each task execution. We compare these resources across 2,826 task pairs, 42 agent–benchmark configurations, and nine datasets.

Token counts include cache traffic exactly once. When the prompt counter includes cache reads, total tokens are prompt plus output; when fresh input, cache reads, cache writes, and output are disjoint, we sum those components. Codex answer-only logs for GPQA-Diamond, AIME, and BeyondAIME supply a CLI counter, which we compare within the same configuration. The completed Claude usage supplement supplies JSON receipts with input, cache, output, and cost fields; each resource mean uses only task pairs with valid receipts.

Figure 9 is the cross-campaign view: each row places the two conditions on a common ratio scale, so horizontal position shows the resource change while the printed values retain native units. The execution cards below unpack the same records by dataset. They are lookup views for the outcome

Table 4: GPQA-Diamond accuracy and execution time comparison between the baseline and + `EpiGraph` settings. For each model, the table reports accuracy and mean per-task wall-clock time in seconds overall and for the three scientific domains—Chemistry, Physics, and Biology. Accuracy changes are in pp; time changes are relative reductions in percent. All models use medium reasoning effort.

	Methods	Setting	Overall (198)		Chemistry (93)		Physics (86)		Biology (19)	
			Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
Codex	GPT 5.6 Sol	-	91.41	246.95	89.25	246.07	95.35	252.11	84.21	227.93
		+ `EpiGraph`	**96.97**	**158.60**	**96.77**	**191.54**	**98.84**	**123.14**	**89.47**	**157.88**
		Δ	↑ 5.56 pp	↑ 35.78%	↑ 7.53 pp	↑ 22.16%	↑ 3.49 pp	↑ 51.16%	↑ 5.26 pp	↑ 30.73%
	GPT 5.6 Terra	-	86.36	189.44	79.57	183.88	95.35	203.62	78.95	152.50
		+ `EpiGraph`	**96.46**	**101.33**	**95.70**	**96.12**	**98.84**	**105.03**	**89.47**	**110.08**
		Δ	↑ 10.10 pp	↑ 46.51%	↑ 16.13 pp	↑ 47.73%	↑ 3.49 pp	↑ 48.42%	↑ 10.53 pp	↑ 27.82%
	GPT 5.5	-	81.82	66.66	70.97	85.38	91.86	46.72	89.47	65.29
		+ `EpiGraph`	**85.35**	**72.56**	**78.49**	**89.42**	**93.02**	**57.12**	**84.21**	**59.86**
		Δ	↑ 3.53 pp	↓ 8.85%	↑ 7.53 pp	↓ 4.73%	↑ 1.16 pp	↓ 22.26%	↓ 5.26 pp	↑ 8.32%
Claude Code	Claude Opus 5	-	83.33	40.60	79.57	54.21	93.02	28.42	57.89	29.19
		+ `EpiGraph`	**96.46**	**40.04**	**96.77**	**54.59**	**96.51**	**25.90**	**94.74**	**32.83**
		Δ	↑ 13.13 pp	↑ 1.38%	↑ 17.20 pp	↓ 0.70%	↑ 3.49 pp	↑ 8.87%	↑ 36.84 pp	↓ 12.47%
	Claude Sonnet 5	-	76.77	30.23	61.29	41.31	94.19	19.86	73.68	22.92
		+ `EpiGraph`	**96.46**	**27.87**	**96.77**	**34.71**	**97.67**	**19.41**	**89.47**	**32.63**
		Δ	↑ 19.69 pp	↑ 7.81%	↑ 35.48 pp	↑ 15.98%	↑ 3.49 pp	↑ 2.27%	↑ 15.79 pp	↓ 42.36%
	Claude Haiku 4.5	-	68.18	110.71	48.39	154.14	89.53	71.42	68.42	75.99
		+ `EpiGraph`	**89.39**	**94.04**	**80.65**	**130.83**	**97.67**	**61.44**	**94.74**	**61.45**
		Δ	↑ 21.21 pp	↑ 15.06%	↑ 32.26 pp	↑ 15.12%	↑ 8.14 pp	↑ 13.97%	↑ 26.32 pp	↑ 19.13%

score, wall-clock time, token count, and matched sample size; they do not introduce a second set of measurements. Panels A–F use the registered medium-effort campaigns, whereas the condensed-matter and biomedical campaigns in panel G use xhigh effort as specified in the experimental setup. A missing token marker indicates unavailable telemetry for that configuration, rather than a zero-valued execution.

The resource comparison separates the outcome effect from the cost of obtaining it. The rows do not show a single uniform overhead: some configurations move the `EpiGraph` marker left while their outcome improves, whereas others spend additional time or tokens to obtain the same kind of gain. This follows from the execution policy: a graph can remove an unproductive branch early, but it can also introduce a check or a repair when an obligation is unresolved. A lower resource ratio therefore indicates avoided work, while a higher ratio alongside a better outcome identifies work spent on an obligation that the baseline left implicit.

The cross-campaign panel must be read within each registered configuration. CMT-Bench and BiomniBench-DA use xhigh effort, while the other scientific and mathematical campaigns use medium effort; their absolute times and token counts are not a common scale for ranking agents. The meaningful comparison is the paired displacement between baseline and +`EpiGraph` within a row. Across these rows, the method changes the execution trajectory without requiring a fixed resource multiplier. This supports transfer of the same graph policy across harnesses and task families while leaving the host's effort setting unchanged.

Table 5: AIME 2024 results by source examination. Accuracy is in percent and time is mean per-task wall-clock seconds. The shaded row reports accuracy changes in pp and relative time reductions in percent; upward arrows indicate improvement. All models use medium reasoning effort.

	Methods	Setting	Overall (30)		AIME I (15)		AIME II (15)	
			Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
Codex	GPT 5.5	-	90.00	58.41	86.67	55.58	93.33	61.24
		+EpiGraph	**100.00**	**56.78**	**100.00**	**55.40**	**100.00**	**58.16**
		Δ	↑ 10.00 pp	↑ 2.79%	↑ 13.33 pp	↑ 0.32%	↑ 6.67 pp	↑ 5.04%
	GPT 5.6 Sol	-	93.33	66.40	93.33	84.05	93.33	48.74
		+EpiGraph	**100.00**	**66.04**	**100.00**	**75.83**	**100.00**	**56.25**
		Δ	↑ 6.67 pp	↑ 0.54%	↑ 6.67 pp	↑ 9.79%	↑ 6.67 pp	↓ 15.41%
	GPT 5.6 Terra	-	96.67	53.32	93.33	64.55	100.00	42.09
		+EpiGraph	**96.67**	**53.77**	**93.33**	**60.98**	**100.00**	**46.56**
		Δ	0.00 pp	↓ 0.84%	0.00 pp	↑ 5.53%	0.00 pp	↓ 10.63%
Claude Code	Claude Opus 5	-	73.33	59.54	80.00	66.83	66.67	52.25
		+EpiGraph	**93.33**	**46.37**	**93.33**	**40.97**	**93.33**	**51.77**
		Δ	↑ 20.00 pp	↑ 22.12%	↑ 13.33 pp	↑ 38.69%	↑ 26.66 pp	↑ 0.92%
	Claude Sonnet 5	-	83.33	70.59	80.00	41.82	86.67	99.37
		+EpiGraph	**93.33**	**57.31**	**93.33**	**43.48**	**93.33**	**71.15**
		Δ	↑ 10.00 pp	↑ 18.81%	↑ 13.33 pp	↓ 3.96%	↑ 6.66 pp	↑ 28.40%
	Claude Haiku 4.5	-	83.33	86.96	86.67	79.90	80.00	94.02
		+EpiGraph	**90.00**	**75.95**	**86.67**	**81.84**	**93.33**	**70.06**
		Δ	↑ 6.67 pp	↑ 12.66%	0.00 pp	↓ 2.43%	↑ 13.33 pp	↑ 25.48%

Table 6: AIME 2025 results by source examination. Accuracy is in percent and time is mean per-task wall-clock seconds. The shaded row reports accuracy changes in pp and relative time reductions in percent; upward arrows indicate improvement. All models use medium reasoning effort.

	Methods	Setting	Overall (30)		AIME I (15)		AIME II (15)	
			Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
Codex	GPT 5.5	-	83.33	95.59	86.67	145.32	80.00	45.85
		+EpiGraph	**100.00**	**68.20**	**100.00**	**80.12**	**100.00**	**56.29**
		Δ	↑ 16.67 pp	↑ 28.65%	↑ 13.33 pp	↑ 44.87%	↑ 20.00 pp	↓ 22.77%
	GPT 5.6 Sol	-	93.33	67.96	93.33	79.82	93.33	56.10
		+EpiGraph	**100.00**	**69.99**	**100.00**	**65.42**	**100.00**	**74.56**
		Δ	↑ 6.67 pp	↓ 2.99%	↑ 6.67 pp	↑ 18.04%	↑ 6.67 pp	↓ 32.90%
	GPT 5.6 Terra	-	70.00	97.17	73.33	86.80	66.67	107.54
		+EpiGraph	**100.00**	**46.68**	**100.00**	**49.83**	**100.00**	**43.54**
		Δ	↑ 30.00 pp	↑ 51.96%	↑ 26.67 pp	↑ 42.60%	↑ 33.33 pp	↑ 59.51%
Claude Code	Claude Opus 5	-	70.00	119.38	66.67	154.84	73.33	83.92
		+EpiGraph	**93.33**	**54.52**	**93.33**	**61.59**	**93.33**	**47.45**
		Δ	↑ 23.33 pp	↑ 54.33%	↑ 26.66 pp	↑ 60.23%	↑ 20.00 pp	↑ 43.45%
	Claude Sonnet 5	-	80.00	84.45	73.33	86.74	86.67	82.17
		+EpiGraph	**86.67**	**61.01**	**80.00**	**67.00**	**93.33**	**55.01**
		Δ	↑ 6.67 pp	↑ 27.77%	↑ 6.67 pp	↑ 22.77%	↑ 6.66 pp	↑ 33.04%
	Claude Haiku 4.5	-	70.00	104.80	66.67	127.11	73.33	82.49
		+EpiGraph	**90.00**	**98.58**	**86.67**	**111.93**	**93.33**	**85.24**
		Δ	↑ 20.00 pp	↑ 5.93%	↑ 20.00 pp	↑ 11.94%	↑ 20.00 pp	↓ 3.32%

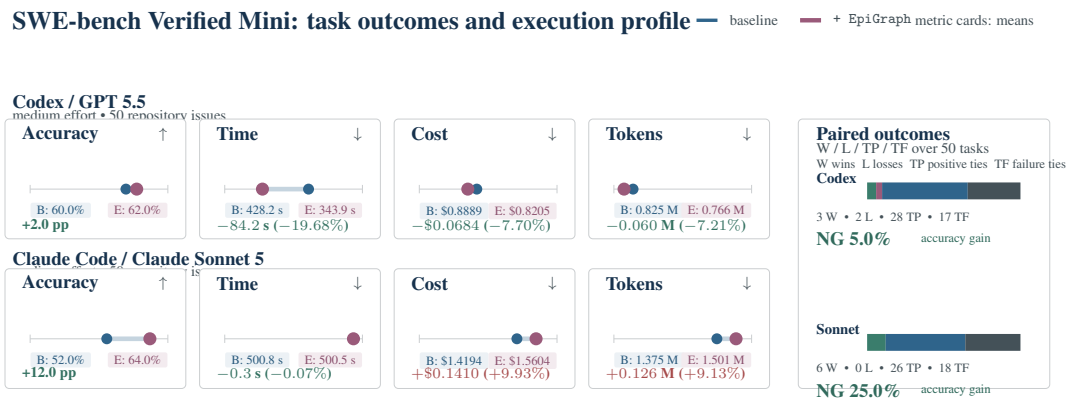


Figure 8: SWE-bench Verified Mini over all 50 repository issues. Each metric card connects the baseline and +EpiGraph means and prints the raw values and Delta. Time is measured in seconds, cost in US dollars, and tokens in millions. The paired-outcome panel reports wins (W), losses (L), ties on positive outcomes (TP), ties on failures (TF), and normalized gain (NG).

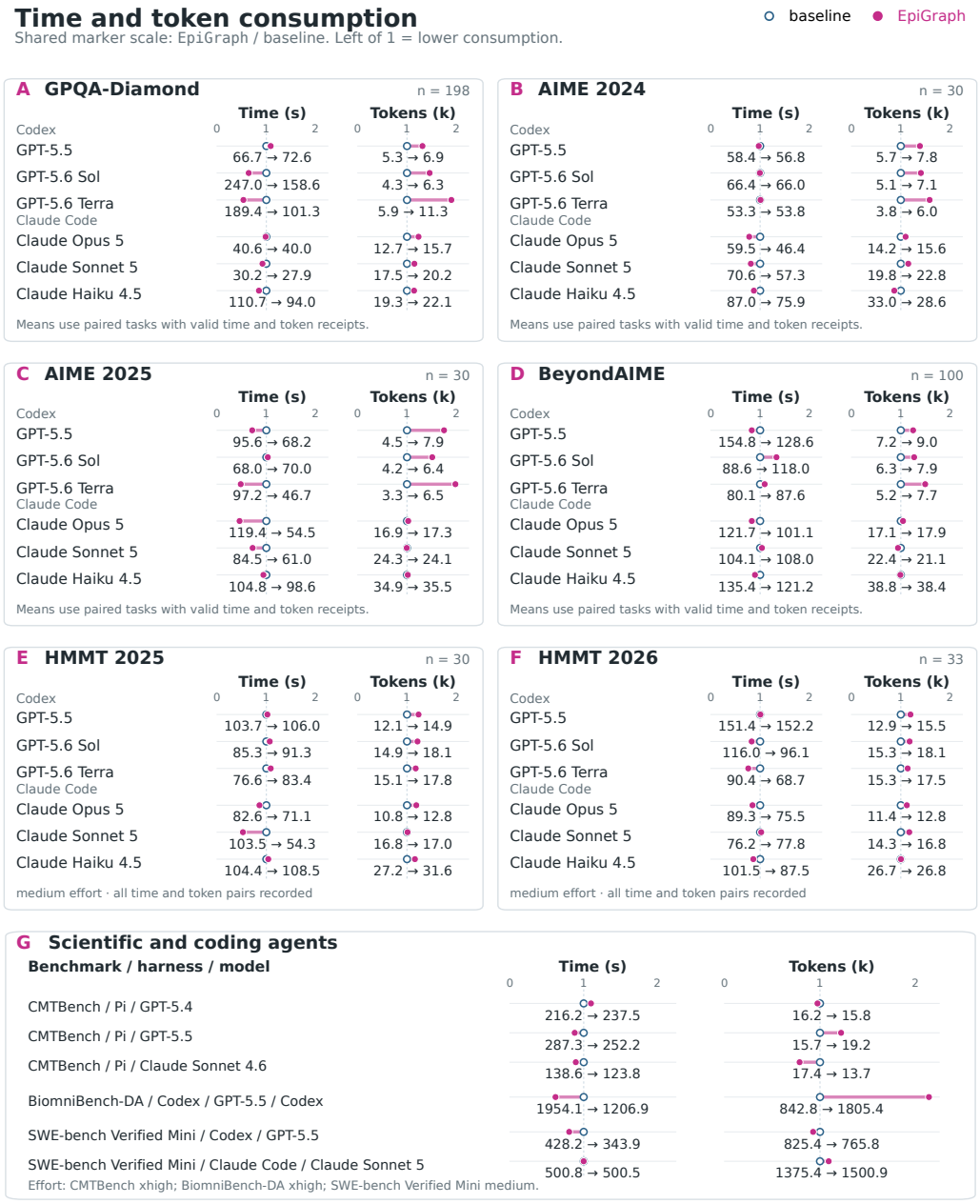


Figure 9: Time and token consumption across nine datasets and 42 configurations. Each row compares baseline (blue open marker) with `EpiGraph` (magenta marker). All marker axes use the same ratio scale: baseline equals 1, and a magenta marker to its left indicates lower consumption. The numbers beneath each pair give baseline → `EpiGraph` mean wall-clock seconds or thousands of tokens. Panels A–F use medium effort. Panel G records CMT-Bench and BiomniBench-DA at xhigh effort and SWE-bench Verified Mini at medium effort, together with matched sample sizes. A dash denotes unavailable token telemetry. Means use the same measured tasks in both conditions for each metric.

E RQ2: PROCESS-LEVEL BIOMEDICAL EVALUATION

The process-level supplement moves from task composition and scoring protocol to task-level comparisons, concrete cases, and saved execution evidence. These materials are different views of the same BiomniBench-DA evaluation and are kept together here.

E.1 RQ2 SUPPLEMENT: PROCESS-LEVEL RUBRIC AND COMPLETION SCORES

We use BiomniBench-DA as the process-level evaluation of LLM agents for real-world biomedical research. We compare scientific execution across its 50-task data-analysis subset (Qu et al., 2026). The questions span oncology (23), metabolic analysis (9), immunology (7), general biology (5), neurology (4), and cardiovascular analysis (2). They require a connected workflow: interpret the question, identify an eligible cohort, compute the requested quantities, evaluate statistical evidence, and explain the biological result. A partially completed workflow can therefore earn rubric credit while leaving a decisive scientific question unresolved.

E.2 GEMINI 3.1 PRO RUBRIC AND SCORING PROTOCOL

Each BiomniBench-DA task is evaluated with its own expert-authored rubric. The rubric specifies the scientific decisions and deliverables required by that task, so the criteria and their point weights can differ across task types. We provide Gemini 3.1 Pro with the task record, the saved execution trajectory, the final answer, and the corresponding rubric. The judge checks the recorded work and answer against every criterion, including the intermediate data objects, analysis choices, quantitative results, interpretation, and supporting sources required by that task (Qu et al., 2026).

The judge first assigns a criterion-level outcome. In the project evaluation records, an "A" outcome receives the criterion's full point weight, "B" receives partial credit, and "C" or a missing required element receives zero; the numeric points remain task-specific because each criterion has its own weight. The task score is the awarded criterion points divided by the total available criterion points, reported on a 0–100 scale. This preserves partial credit for a workflow that completes useful scientific work while omitting a required decision or deliverable. A missing trajectory or answer is assigned zero for the corresponding benchmark entry.

The six dimensions shown in the main-text overview are a cross-task process profile constructed from the criterion metadata. They are not six additional questions asked by Gemini and they do not replace the task-native rubric. We map each criterion to one of data handling, method selection, statistical rigor, biological interpretation, scientific reasoning, or source reliability, normalize its awarded points to the criterion's available points, and average the mapped criteria over the tasks in which that dimension is defined. For the case-level profile used in the intervention analysis, the stored A/B/C levels are read as 100/50/0, with missing criteria also assigned 0; a dimension with no task-specific criterion is reported as zero for that case. This gives the aggregate profile a common scale while preserving the original criterion weights for the task score.

The remaining summaries use the task scores produced by this protocol. The mean averages all 50 task scores, the median identifies the middle task, the non-zero count records tasks receiving any rubric credit, and a threshold curve counts tasks whose score reaches the specified completion level.

Following the benchmark taxonomy, we summarize this public subset along two orthogonal axes: disease category and analytical task type. The 50 tasks cover six disease categories and 16 of the benchmark's 17 task types. Figure 10 shows the intersection matrix, with marginal totals printed above the columns and beside the rows. The labels and counts are computed directly from the task metadata used in the evaluation. This view describes benchmark composition, while the score analyses below use the same 50 task IDs.

BiomniBench-DA assigns one hour to each task, and all systems are scored on the benchmark's 50-task denominator. The GPT 5.5 group contains Biomni (Huang et al., 2025), Codex, Codex + `EpiGraph`, Pi, and Pi + `EpiGraph`. The additional systems are EvoScientist[17], OpenScience (Syn-

[17] https://github.com/EvoScientist/EvoScientist

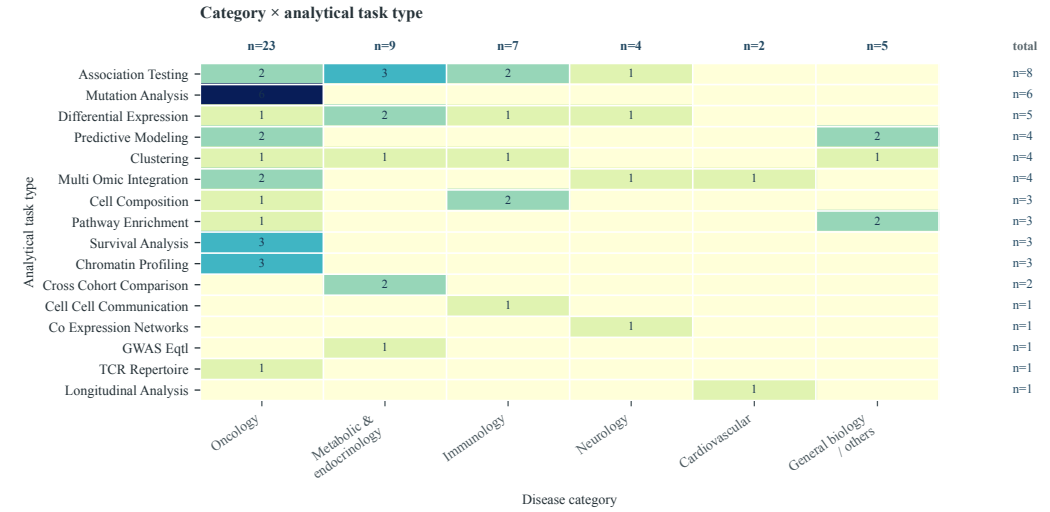


Figure 10: Composition of the 50-task public BiomniBench-DA subset. The matrix shows the intersection of disease category and analytical task type; numbers inside cells are task counts, while the marginal totals are printed above the columns and beside the rows. Blank cells contain no task.

thetic Sciences)[18], OpenScience (AI4S Research)[19], and Wisp Science[20]. The parenthetical names identify the organizations behind the two Open Science projects. The released BiomniBench-DA evaluations use DeepSeek V4 Pro as the agent model[21]. OpenScience describes a research agent with an integrated workbench; the AI4S Research project, now named Open Science Desktop, and Wisp Science describe local-first research workbenches (The Open Science Desktop Contributors, 2026; Xu et al., 2026b). Together these systems represent both general coding agents and environments organized specifically for scientific work.

To distinguish partial completion from reaching a specified standard, let $s_{m,i} \in [0, 100]$ be the Gemini 3.1 Pro rubric score for system m on task i. We report the mean rubric score and the fraction of tasks reaching each threshold $\tau \in \{0.1, 0.2, \ldots, 1.0\}$:

$$\bar{s}_m = \frac{1}{50} \sum_{i=1}^{50} s_{m,i}, \qquad \mathrm{Pass}_m(\tau) = \frac{100}{50} \sum_{i=1}^{50} \mathbb{I}[s_{m,i} \geq 100\tau]. \tag{23}$$

Thus pass@0.7 is the percentage scoring at least 70 points, and pass@1.0 is the percentage receiving full credit. The median describes the middle task, while the non-zero count records how many tasks receive some rubric credit. The additional-system rows use the released Gemini score for each task; all nine rows retain the complete 50-task denominator.

Figure 11 reports rubric scores over all 50 task IDs. Missing baseline outputs receive the benchmark score of zero. Codex + `EpiGraph` reaches 67.48 points, compared with 59.16 for Codex and 48.92 for Biomni: improvements of 8.32 and 18.56 rubric points. The highest mean among the four additional scientific systems is EvoScientist's 65.26; Codex + `EpiGraph` exceeds it by 2.22 points. Codex + `EpiGraph` has 49 non-zero scores and nine full-credit scores; Codex has 43 and five. Pi + `EpiGraph` reaches 53.38 points versus Pi's 51.14, and increases the number reaching 70 points from 15 to 19. The two host agents therefore provide complementary views of the same execution policy.

The score distributions and threshold curves locate where additional credit changes the practical completeness of an answer (Figure 11). At 40 points, Codex + `EpiGraph` completes 45 tasks versus Codex's 36, a gain of 18 percentage points. At 50 points the counts are 39 and 33, a gain of 12 percentage points. Both reach 70 points on 24 tasks, while full-credit answers increase from five to

[18] https://github.com/synthetic-sciences/openscience
[19] https://github.com/ai4s-research/open-science
[20] https://github.com/xuzhougeng/wisp-science
[21] https://github.com/omicverse/BiomniBench-AI4S

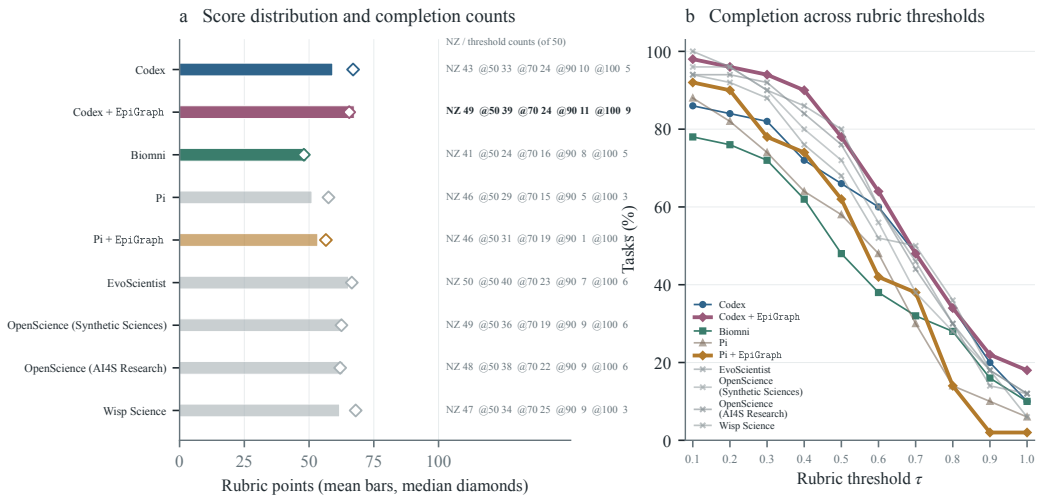


Figure 11: BiomniBench-DA completion summary over all 50 tasks. (a) Mean bars, median diamonds, and the counts of non-zero and threshold-reaching scores for all nine systems. (b) The corresponding ten threshold curves. Codex + `EpiGraph` is highlighted in purple and Pi + `EpiGraph` in gold; Biomni and Codex use green and blue, respectively, and the remaining systems use muted tones.

nine. Wisp Science has the largest count at 70 points (25), while Codex + `EpiGraph` has the largest full-credit count (nine) and mean score. The curves keep these different forms of completion visible together.

E.3 Detailed Case Score Atlas

The main-text Figure 4 gives the aggregate comparison. The atlas below retains only the case-level scores that are needed to select and interpret the six trace walkthroughs; the task-composition matrix above is the single taxonomy display used in this appendix.

The atlas is used to select cases by scientific role rather than by score alone. The six columns cover a rule-defined subgroup, a multiple-testing decision, a paired measurement analysis, a signed relation, a survival comparison, and a genomic subtype synthesis. The variation across systems is therefore diagnostic: a system can match a baseline on a simple count while losing credit when the task requires a denominator, a correction rule, or a calculation to be carried into the final interpretation. The detailed walkthroughs trace these differences to the prerequisites, expected observations, and receipts that the rubric scores. The atlas thus supplies the selection frame for the case studies, while the case studies explain why the scores differ and how the graph connects an intermediate scientific object to the reported claim.

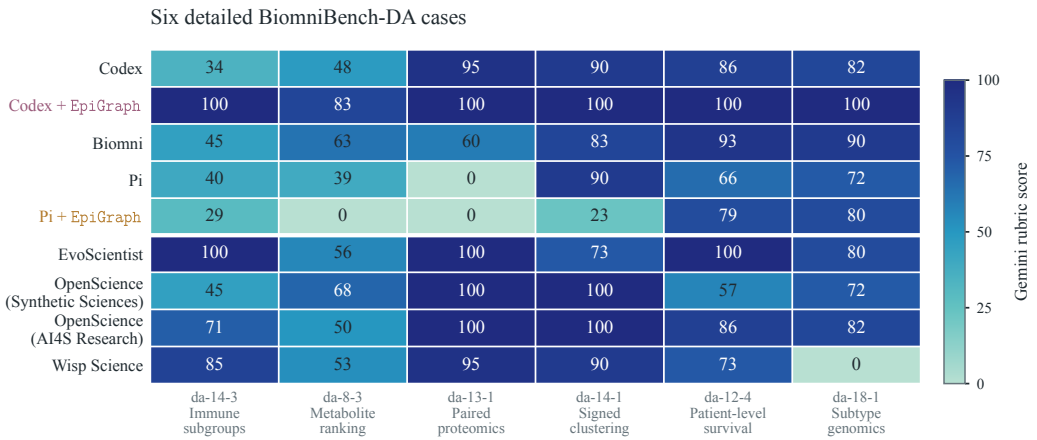


Figure 12: Rubric scores for the six detailed BiomniBench-DA cases. Columns correspond to immune subgroups, metabolite ranking, paired proteomics, signed clustering, patient-level survival, and subtype genomics. The rows retain the nine systems used in the main-text comparison, and the shared color scale runs from zero to 100 points. These six columns are the entry points for the trace-level walkthroughs and saved scientific artifacts in Appendix E.5.

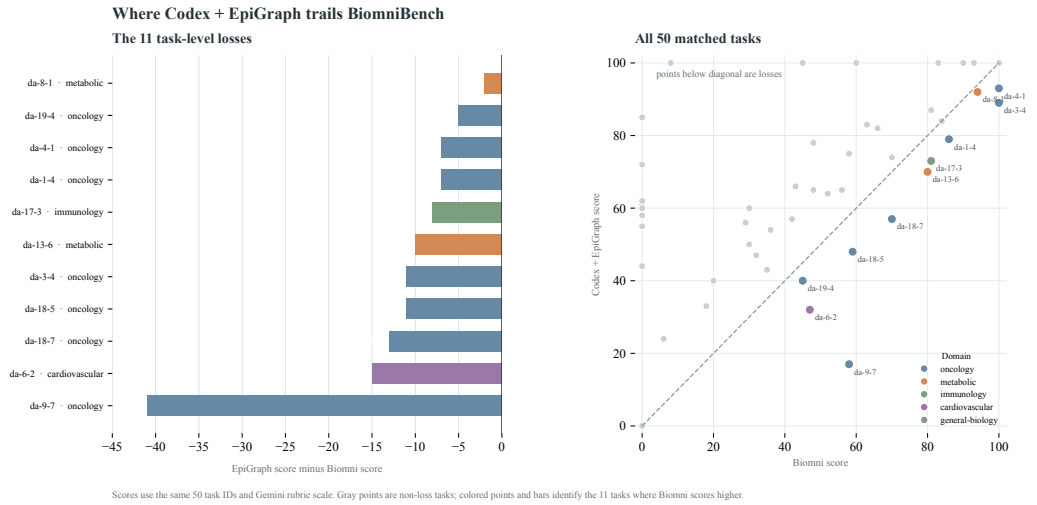


Figure 13: Task-level losses of Codex + `EpiGraph` against BiomniBench. The left panel ranks the 11 losses by the score difference `EpiGraph` − Biomni; the right panel shows all 50 matched task IDs, with losses highlighted by domain and non-loss tasks in gray. Both panels use the released Gemini rubric scores and the same task-key join.

E.4 RQ2 Supplement: Process-Level Boundary Cases

The appendix failure analysis compares the recorded Codex + `EpiGraph` scores with the public Biomni scores on the same 50 BiomniBench-DA task IDs. This is a system-level comparison: a loss is a task on which the Biomni rubric score is higher than the score of Codex + `EpiGraph`. The comparison contains 34 wins, 11 losses, and 5 ties for Codex + `EpiGraph`. Figure 13 shows both views at once: the left panel ranks every loss by its score gap, and the right panel places all 50 matched tasks against the equality diagonal.

The largest gap is concentrated in `da-9-7`; the remaining losses are closer, with nine of the other ten gaps no larger than 15 points. Seven of the 11 losses are oncology tasks, while the remaining four span cardiovascular, metabolic, and immunology analyses. Association testing and mutation analysis each account for three losses; the other five losses are distributed across cross-cohort com-

parison, differential expression, chromatin profiling, clustering, and longitudinal analysis. Relative to the Codex baseline, `EpiGraph` still improves four of these Biomni losses (`da-13-6`, `da-18-5`, `da-4-1`, and `da-6-2`) and ties one, while it declines on six. The six within-Codex declines identify the sharper cases for trace inspection. The figure keeps the score gaps and their associated obligations visible at the task level, including cases where several upstream issues contribute to one incomplete biological conclusion.

The public Biomni comparison supplies task scores and task metadata, whereas the criterion-level records in this project describe the Codex baseline and Codex `+EpiGraph` traces. The trace evidence identifies the path from a technically valid analysis to a sufficiently specific biological explanation. Scope choices, statistical thresholds, candidate selection, and missing checks are therefore read as points along the same interpretation chain, and the six detailed walkthroughs analyze prerequisites, actions, observations, and receipts directly.

E.5 Concrete Process-Level Case Performance on BiomniBench-DA

This section explains the data, decisions, and verification outputs behind the biomedical execution graphs. We use a prerequisite to mean a fact that must be established before a calculation is meaningful, an expected observation to mean a checkable property of its output, and a receipt to mean the recorded output of the executed command. For example, "one assignment per patient" is an expectation; a table containing exactly 1,303 unique patient assignments is a receipt. The figures visualize the prerequisite, action, and receipt dependencies evidenced by the execution event streams. Biomni and Codex are the baseline conditions; `EpiGraph` is the Codex condition augmented with our method. The six cases expose different recoverable scientific intermediates: a classification rule, a test family, a time label, a signed relation, an analysis unit, and a calculation-to-answer dependency.

To make the visual grammar explicit before the six trace walkthroughs, Figure 14 gives the abstract execution meta-graph used to read them. It is drawn from the same objects that appear in each case-specific trace: the task contract is the root; prerequisite, action, expected-observation, and receipt nodes are placed according to their dependency structure; solid labeled arrows record `Guards`, `Predicts`, and `Verifies` relations; and dashed arrows with operator badges mark graph updates. The drawing provides a shared reading grammar. Each case below instantiates this grammar with the nodes, branches, revisions, and scientific outputs present in its own recorded trace.

E.6 Immune Subgroups: Define the Rule Before Counting

To read the execution meta-graph, start with the unit of the question. A patient can contribute several samples, and two samples from the same patient can have different immune scores. Counting sample labels answers a different question from counting patients. Moreover, an existing subgroup column contains the output of a labeling procedure, while the continuous score columns allow the procedure itself to be checked. The `Compile` operation makes these dependencies explicit: define the eligible population, verify the labeling rule, assign one label per patient, and report four proportions that form a complete partition.

To expose the labeling decision, denote a patient's two scores by z_M and z_L. For this dataset the reconstructed high indicators are $h_M = \mathbb{I}[z_M > 1.65]$ and $h_L = \mathbb{I}[z_L > 1.65]$. The pairs $(0, 0)$, $(1, 0)$, $(0, 1)$, and $(1, 1)$ correspond respectively to balanced, myeloid, lymphoid, and system-wide dysregulation. The high direction matters: a large negative score is still low under this rule. The threshold sets the membership of every category and therefore guards every subsequent count. This is why validating it is more informative than merely reading the four names from an existing column.

To verify the proposed rule, the recorded execution first evaluates the candidate cutoff 1.6448536 and obtains three myeloid and five lymphoid disagreements with the supplied high/low labels. It then evaluates 1.65 and obtains zero disagreements on both axes. Reconstructing the four subgroups from the supplied high/low flags also yields zero disagreements. The recorded commands test the predicates; the receipt supplies their outcomes. `Revise` changes the contradicted cutoff, and `Admit` allows the checked rule to be used for patient-level aggregation. The graph places the reason to proceed on the verified dependency between rule and action.

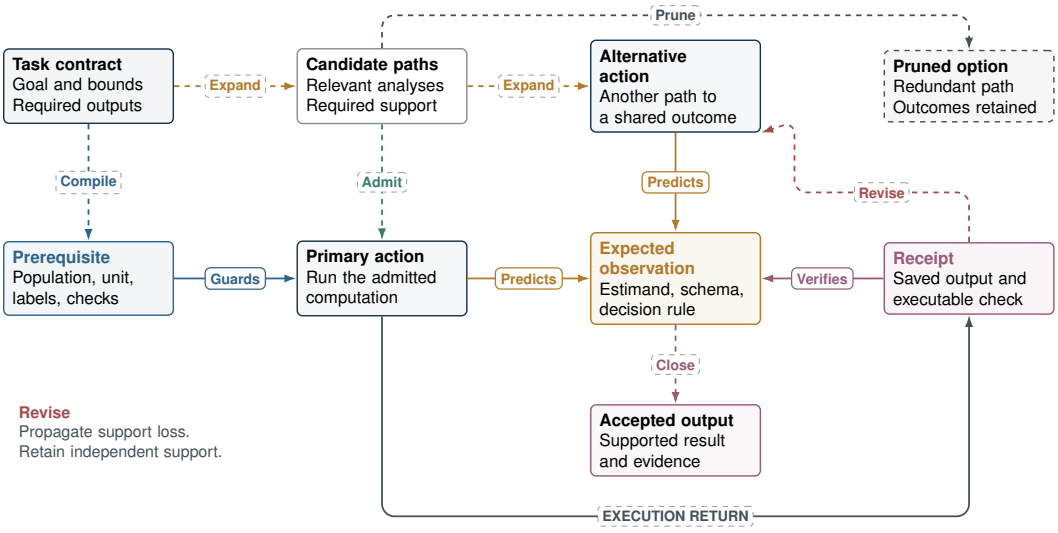


Figure 14: Detailed execution meta-graph used to read the case studies. Boxed labels denote the named edge types; dashed operator arrows denote graph updates. Colors match the case-study graphs and method notation.

Figure **??** gives a compact paired view of the same task: the upper strip records the Codex baseline sequence, while the lower strip follows the `EpiGraph` execution through rule checking, a `Revise` update, patient-level aggregation, and partition closure. The complete dependency graph, including the auxiliary denominator and repair branches, follows below.

To resolve duplicated measurements, the agent selects the 1,320 baseline severe rows and groups them by patient identity. Seventeen patients appear twice, so the unique-patient denominator is $1{,}320 - 17 = 1{,}303$. Four of those patients have conflicting row labels. The analysis averages each continuous score within a patient and then applies the threshold. This makes the final assignment depend on the measured scores and avoids making the accession ordering decide which label wins. Once every patient has one subgroup, the counts are 122 balanced, 73 myeloid, 314 lymphoid, and 794 system-wide.

To check closure, let n_g be the count in subgroup g and N the number of eligible patients. The reported percentage is $100n_g/N$. Here $122 + 73 + 314 + 794 = 1{,}303$, and the corresponding percentages sum to 100%. These simple identities are useful evidence: the first checks that the entire population was assigned once, and the second checks that the answer uses a common denominator. The largest group is system-wide at 60.94%; the smallest is myeloid at 5.60%. Both statements can be traced to the same verified table.

A Execution meta-graph

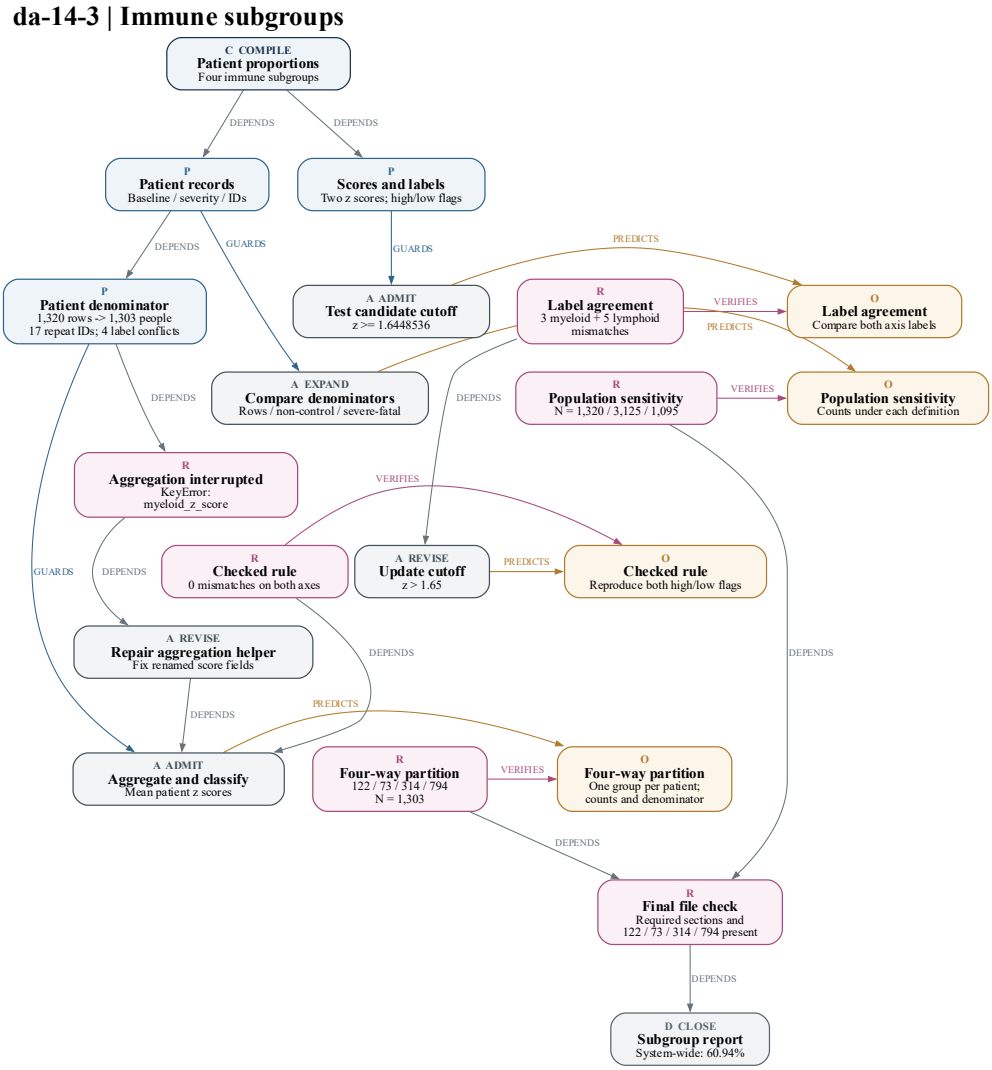


Figure 15: Execution meta-graph for da-14-3. The graph connects the cutoff mismatch and the resulting `Revise` update with patient aggregation, the primary count receipt, and the independently computed denominator sensitivities. An aggregation field error triggers a helper repair before the successful rerun. Final file checks connect the computed partition to report closure.

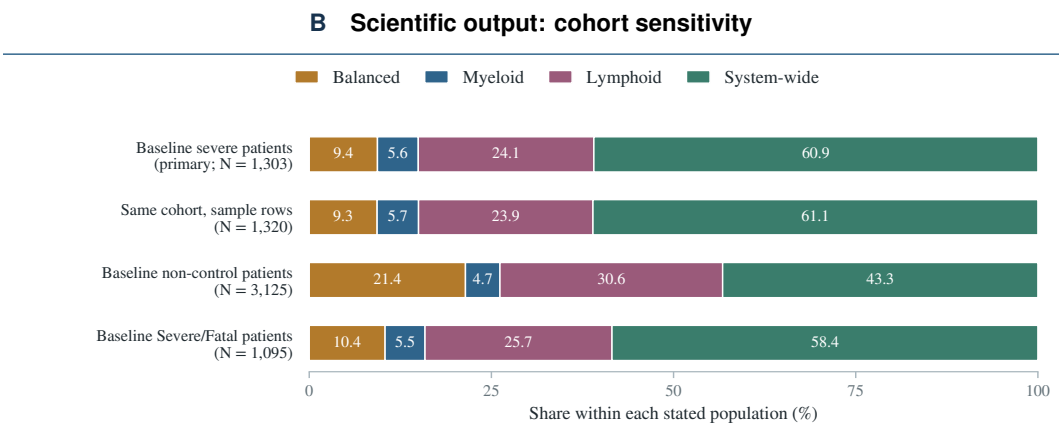


Figure 16: Population definition has a larger effect than duplicate-row counting in da-14-3. All four bars come from sensitivity calculations within the same recorded `EpiGraph` run. The primary cohort is baseline severe unique patients. The broad non-control cohort includes non-severe or unknown-severity patients; the Severe/Fatal grade filter excludes severe records with missing finer grades. Segment labels give percentages, and each row states its own denominator.

To understand which choice changes the scientific answer most, Figure 16 varies one definition at a time. Counting all 1,320 sample rows under the same primary filter yields a system-wide fraction of 61.06%, only 0.12 percentage points above the patient-level 60.94%. Broadening eligibility to 3,125 baseline non-control patients yields 43.26%, a decrease of 17.67 percentage points. Requiring the finer Severe/Fatal grade yields 1,095 patients and 58.45%; that filter omits 208 baseline rows already marked severe in the broader severity field. The large difference comes from whom the analysis includes. The graph exposes that cohort choice before the downstream percentage is accepted.

To connect these decisions to the baselines, their traces show that both Biomni and Codex inspect missingness and repeated patients and then count the precomputed subgroup column. Their rubric evaluations specifically identify the omitted threshold reconstruction. The execution graph and paired trace show where this criterion is discharged. `EpiGraph` also explains the category ordering, threshold assumptions, and denominator choices, completing the remaining reporting and interpretation criteria. The case therefore links three levels of evidence: the event that checks the rule, the patient table produced under that rule, and the criterion that rewards the completed analysis.

The comparison analyses make both the shared capability and the added precision visible. EvoScientist also reconstructs a cutoff near 1.65 and receives full credit; its primary proportions describe 3,142 non-healthy baseline rows. OpenScience (Synthetic Sciences) counts 2,203 unique infected baseline patients, while OpenScience (AI4S Research) uses 1,144 infected severe patients. Wisp Science likewise centers its analysis on infected patients. These are explicit but different population definitions. The `EpiGraph` record connects its chosen severe-patient population to an exact threshold test, resolves repeated patients using continuous scores, and closes the four-group partition against a common denominator. The sequence from eight axis-label disagreements to zero, followed by 1,303 unique assignments, is a particularly direct example of `Revise`, `Admit`, and `Close` acting on machine-checkable scientific objects.

E.7 Metabolomics: A Complete Result Can Contain Zero Discoveries

To read the execution meta-graph, the task asks for the ten metabolites whose abundances differ most between two glucose-response groups and for lipid species that statistically mediate their clinical differences. A metabolite is a measured small molecule; a feature is one measured table entry. Ranking features and establishing discoveries are separate operations. A ranking always has a first ten entries, whereas a discovery requires sufficient statistical evidence after accounting for how many features were tested.

A Execution meta-graph

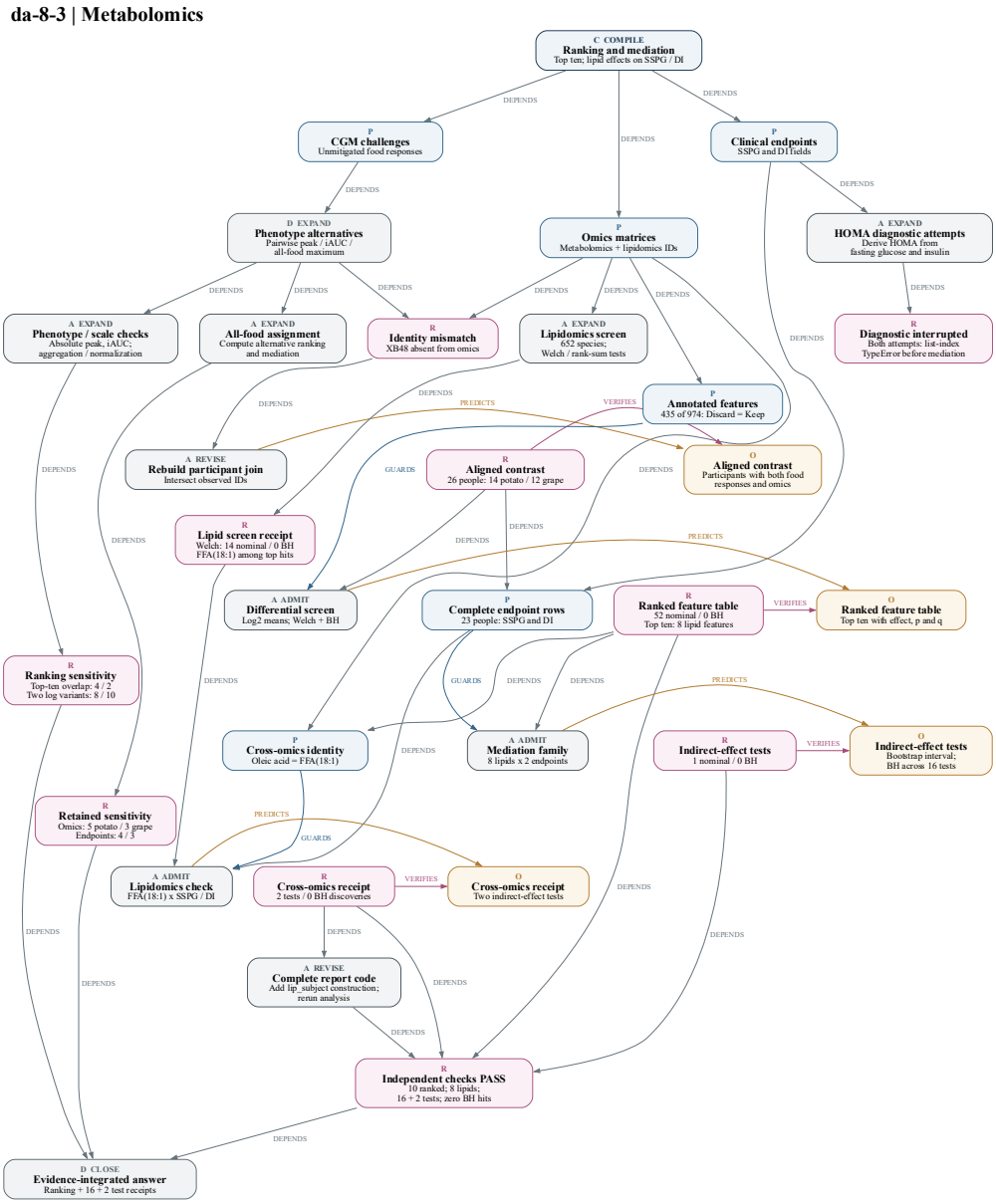


Figure 17: Execution meta-graph for da-8-3. Participant-identity repair supports the metabolite screen and 16 mediation tests. A 652-species lipid screen and chemical identity support two cross-omics checks. Phenotype and scale sensitivities branch separately; the all-food contrast is retained. Both HOMA diagnostic attempts stop at a list-index error. Report-code repair and independent checks close the primary result families.

To establish the correct analysis unit, `EpiGraph` identifies 113 omics sample columns representing 38 participant IDs. These columns include replicate measurements of the same person. Averaging log-abundances within a person prevents someone with more technical replicates from receiving more weight. Matching omics participants to glucose measurements leaves 26 people in the contrast, with 14 potato-spikers and 12 grape-spikers under the reported classification. The feature eligibility flag retains 435 of 974 metabolomics features. The prerequisites for testing are now explicit: aligned people, one summary per person, and a fixed eligible feature set.

To interpret the test output, a raw p-value assesses one feature's evidence against its null hypothesis. Testing hundreds of features creates many opportunities for small raw p-values, so the recorded analysis applies Benjamini–Hochberg (BH) false-discovery-rate correction (Benjamini & Hochberg, 1995). It returns 52 nominal results with $p < 0.05$ and zero with adjusted $q < 0.05$ across the 435 tests. `EpiGraph` reports the top ten with their names, effect sizes, raw p-values, and adjusted q-values. The ranking contract is satisfied and the corrected discovery count is zero.

To close the mediation branch, the agent takes the eight lipid-class entries among those ten metabolites and tests each against two clinical endpoints, giving $8 \times 2 = 16$ primary tests. The endpoints summarize insulin resistance and beta-cell function; complete endpoint measurements are available for 23 participants. A mediation calculation estimates whether the group difference is associated with an indirect path through a candidate lipid. The recorded output gives one nominal example, tauroursodeoxycholic acid for the insulin-resistance endpoint, with $p = 0.0449$ but adjusted $q = 0.6272$. None of the 16 tests passes the corrected threshold. The graph closes both the candidate description and the final statistical verdict.

Figure 18 exposes the intermediate result behind that distinction. Each feature has an effect-size bar and two connected statistical values. Correction moves all ten beyond the 0.05 threshold; the feature names and ranked effects remain reportable. The graph can therefore verify a completed ranking and a zero-discovery inference together.

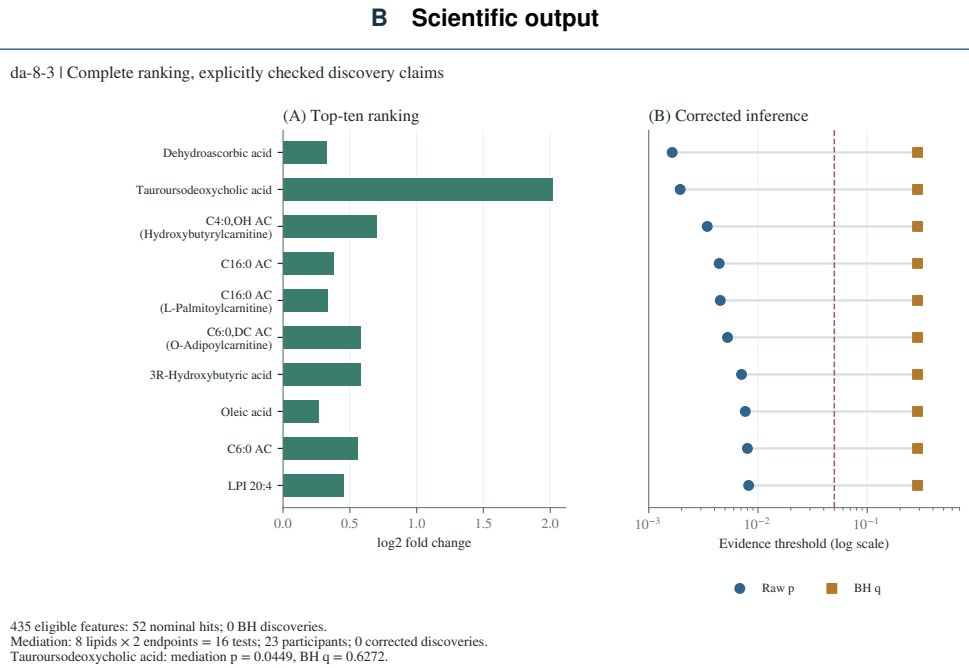


Figure 18: The recorded metabolomics run separates ranked candidates from corrected discoveries. The graph connects subject alignment to the two acceptance conditions. Left: saved $\log_2$ fold changes for the top ten features. Right: raw p and BH-adjusted q for those same features; the dashed line marks 0.05. All plotted q values lie above the threshold. The mediation branch has its own 16-test family and also returns zero corrected discoveries. The two C16:0 entries retain their distinct source annotation strings.

To explain the score gain precisely, the Codex baseline also applies BH correction and reports no corrected mediation discoveries. The observable advantage is the completeness of the connected outputs: `EpiGraph` supplies lipid classes and identifiers, links reported metabolites to clinical endpoints, and reports the associated test families. Relative to Codex, its rubric gains are +4 in phenotype construction, +11 in top-ten annotation, +6 in phenotype association, +9 in integration/mediation, and +5 in statistical treatment, totalling +35 points. These gains connect `Compile` and `Close`: each requested analytical component has an explicit output to verify.

The additional traces locate the advantage earlier in the analysis as well (Figure 29a). EvoScientist assigns phenotypes using the food with maximum incremental glucose area. After matching to the omics measurements, its differential test contains 19 potato-group columns from four people and four grape-group columns from one person. Those columns enter the test as independent observations. Repeated measurements refine a person's abundance estimate; treating them as separate people gives that person repeated influence over the group comparison. The mediation branch then returns to the participant level and has only four endpoint-complete people. The `EpiGraph` execution establishes a common participant-level representation before ranking or mediation, so the biological unit remains consistent across both branches.

The identity of the contrast is a second dependency. OpenScience (AI4S Research) Research) finds no grape-spikers in the omics overlap under its maximum-food classification and proceeds with four potato-spikers versus 15 rice-spikers. Its subsequent tests therefore describe a different food comparison. `EpiGraph` reconstructs the requested relative potato-versus-grape response from unmitigated glucose measurements and records the resulting 14/12 partition. OpenScience (Synthetic Sciences) also retains a relative potato-versus-grape comparison, using peak glucose and a 15/11 partition. It tests all 974 features and extends mediation to additional endpoints and predictor variants, with a suggestive Sobel threshold of 0.20. Wisp Science uses a normalized food-response classification, seven participants, and 15 mediation tests, reporting one bootstrap interval excluding zero. These records make the premise, sample set, and decision threshold inspectable alongside every reported molecular association.

The final advantage is the precision of closure. The `EpiGraph` receipt checks that the ranked list contains exactly ten named entries, that eight are lipid-class features, that the primary mediation family contains exactly 16 rows, and that its corrected discovery count is zero. It also retains an orthogonal lipidomics check: oleic acid maps to FFA(18:1), with no supported mediation for either endpoint. This separates molecular identity, clinical association, and cross-assay confirmation into distinct evidence obligations. `Compile` creates those obligations, `Admit` requires aligned participants and eligible features, and `Close` accepts the answer only after the corresponding saved outputs reconcile.

E.8 Proteomics: Close Both the Set Comparison and the Statistical Test

To read the execution meta-graph, da-13-1 asks which plasma proteins change after six months of feminizing gender-affirming hormone therapy and whether changes differ between cyproterone acetate (CPA) and spironolactone (SPIRO) regimens. The assay table measures normalized protein abundance at baseline and follow-up. A paired comparison subtracts a person's baseline value from their own follow-up value, reducing the influence of stable between-person differences. It therefore requires the identity and time label of both measurements.

To admit the paired analysis, `EpiGraph` constructs an $80 \times 5{,}420$ sample-by-assay matrix with zero missing values and removes two bridge-replicate rows from the participant selection. The metadata-defined pairing produces 20 complete CPA pairs and 16 complete SPIRO pairs. For assay j, the analysis uses the change $\Delta_{ij} = y_{ij,6m} - y_{ij,0m}$ for participant i. It tests changes within CPA, within SPIRO, across the pooled cohort, and between regimens, applying BH correction separately within each family of 5,420 assays. The analysis yields 265 CPA discoveries, 73 SPIRO discoveries, 539 pooled discoveries, and seven between-regimen interaction discoveries.

To distinguish the two comparison contracts, let A and B be the assay sets with $q < 0.05$ within CPA and SPIRO. The shared set has $|A \cap B| = 27$ assays. Consequently, CPA-only has $265 - 27 = 238$, SPIRO-only has $73 - 27 = 46$, and their union has $265 + 73 - 27 = 311$ assays. These three set sizes are arithmetic derived from the recorded receipt. A set difference asks whether two significance lists overlap; an interaction test asks whether the measured changes themselves differ between regimens.

A Execution meta-graph

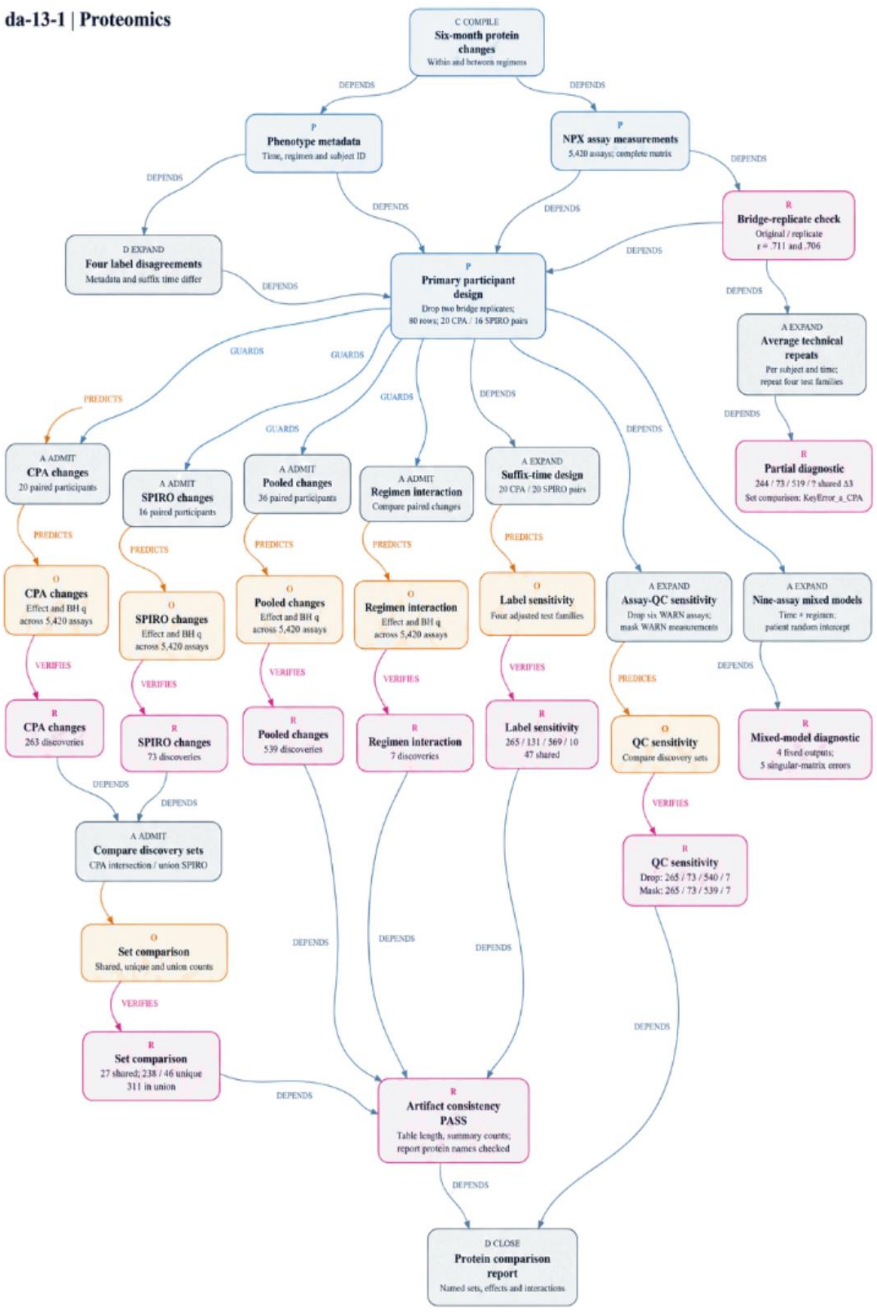


Figure 19: Execution meta-graph for da-13-1. The paired matrix supports four statistical families and a shared/unique set comparison. Separate branches evaluate sample labels, assay QC, technical replicates, and mixed models. The nine-assay mixed-model loop returns four fits and five singular-matrix errors. Technical-replicate averaging produces counts before a set-comparison column lookup fails. Final read-back assertions check table length, discovery counts, and protein-name presence in the report.

An assay can pass a threshold in one list and miss it in the other without passing a direct between-regimen test. `EpiGraph` therefore closes both outputs explicitly.

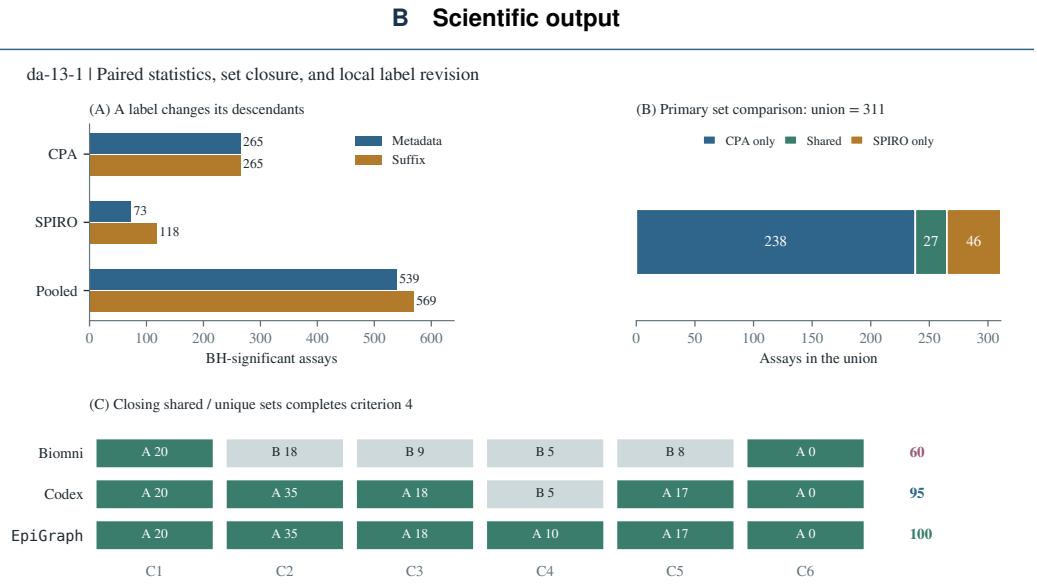


Figure 20: Three scientific views of the proteomics output. Changing four time labels affects SPIRO and pooled discoveries while the CPA count stays fixed; the primary receipt yields 238 CPA-only, 27 shared, and 46 SPIRO-only assays; and the rubric cells show the completed criterion values.

To locate the benefit relative to an already capable baseline, Codex scores 95 and its rubric credits its paired tests, correction, and interaction analysis. The remaining five-point gap is the missing explicit shared/unique set comparison. Biomni scores 60; its evaluation additionally identifies incomplete regimen-specific reporting. `EpiGraph` receives full marks by recording the within-regimen counts, their 27-assay intersection, and the seven interaction results. This example shows how a graph can improve an answer near completion: the unresolved deliverable remains a live dependency until its own receipt is present.

Figure 20 places the scientific intermediates beside the evaluation: discovery counts under two label policies, the primary shared/unique partition, and the criterion-level three-system comparison. This links the five-point completion gain to an explicit set operation that readers can recompute from the printed counts.

A biological effect depends on the question asked of it. The saved protein-level results let us examine the scientific consequence of maintaining separate evidence obligations. Normalized protein expression (NPX) is measured on a $\log_2$ scale; a positive change means that the assay signal increases between baseline and six months. For protein j, write the mean paired changes as $\overline{\Delta}_{j,\mathrm{CPA}}$ and $\overline{\Delta}_{j,\mathrm{SPIRO}}$. A within-regimen test asks whether one mean change is zero. The direct comparison asks whether the two mean changes are equal. A difference between two significance labels alone does not answer that second question (Gelman & Stern, 2006). The graph therefore creates separate expected observations for the two within-regimen tests and the between-regimen comparison.

Figure 21a displays the complete assay landscape. The seven corrected between-regimen results are CFC1, EDDM3B, INSL3, SPINT3, CXCL13, NELL1, and PRL. INSL3 decreases under both regimens, with mean changes of -5.345 and -1.775 NPX and interaction $q = 0.00161$. Here the between-regimen result concerns the magnitude of a shared direction. NELL1 exposes a different pattern: its CPA mean is $+0.206$ NPX and its SPIRO mean is -0.251 NPX. Pooling the 20 CPA and 16 SPIRO pairs yields

$$\overline{\Delta}_{\mathrm{NELL1,pooled}} = \frac{20\overline{\Delta}_{\mathrm{NELL1,CPA}} + 16\overline{\Delta}_{\mathrm{NELL1,SPIRO}}}{36} \simeq 0.00287 \text{ NPX}. \tag{24}$$

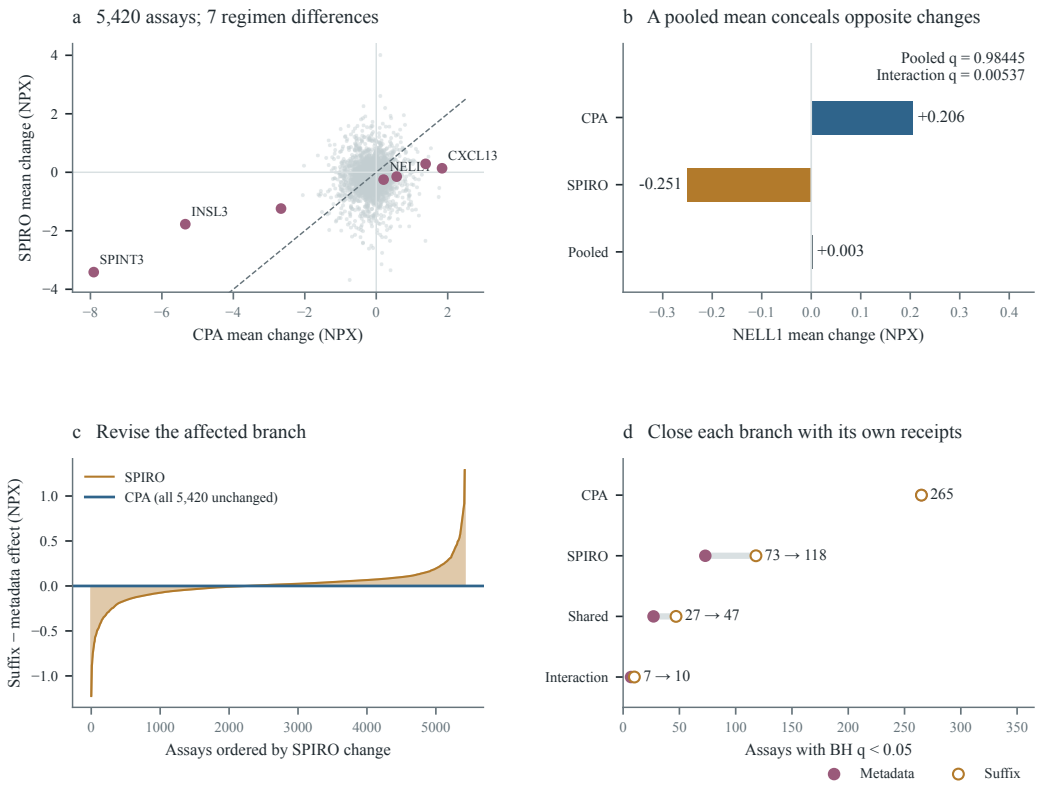


Figure 21: Scientific consequences of explicit evidence obligations in paired proteomics (da-13-1). (a) Mean paired protein changes under CPA and SPIRO for all 5,420 assays; magenta marks the seven assays with interaction $q < 0.05$. The dashed diagonal denotes equal mean changes. (b) NELL1 has oppositely directed regimen means and an almost zero pooled mean, while its direct comparison meets the corrected threshold. (c) Changing four disputed time labels changes the SPIRO mean effects; CPA means and all CPA test columns remain identical across the 5,420 rows. (d) Each label interpretation retains its own corrected discovery counts. Every value comes from the saved primary and sensitivity result tables.

The pooled test has $q = 0.98445$, whereas the direct between-regimen comparison has $q = 0.00537$ (Figure 21b). The two receipts answer different biological questions: whether the combined cohort changes on average, and whether the change differs by regimen. Keeping both obligations in the graph preserves a regimen-associated pattern that is obscured in the pooled mean.

The label sensitivity analysis then tests the dependency structure itself. Four SPIRO sample names disagree with their metadata about measurement time. The primary interpretation uses the metadata; the alternative uses the time suffix. With the alternative labels, SPIRO's complete pairs increase from 16 to 20, its discoveries from 73 to 118, the shared discoveries from 27 to 47, and the between-regimen discoveries from seven to ten. In contrast, the CPA pair counts, mean changes, raw p-values, and corrected q-values are identical in every one of the 5,420 assay rows. Panels c and d make this dependency visible at both the molecular and summary levels: the changed premise affects the quantities that depend on it, and each interpretation retains a complete result.

These artifacts give the graph edges an operational meaning. A `Guards` edge connects sample-derived subject identifiers and the adopted time-label rule to paired subtraction. `Predicts` edges connect the analysis action to expected assay-indexed effects, corrected test families, and set memberships. `Verifies` edges associate each receipt with its acceptance predicate: the statistical tables supply assay-level effects and corrected tests; the read-back assertions check table length, summary counts, and protein-name presence in the report. `Revise` changes a time-label premise, and the sensitivity branch records its consequences for the paired analysis. `Close` checks the resulting reporting

obligations. The representation carries the scientific contract across molecular effects, alternative labels, and final interpretation.

The additional proteomic traces clarify how this capability relates to specialized scientific systems. EvoScientist and OpenScience (AI4S Research) Research) also recognize the conflicting SPIRO time labels and recover 265 CPA and 73 SPIRO discoveries under the metadata interpretation. OpenScience (AI4S Research) explicitly compares paired changes and reports seven corrected between-regimen differences; Wisp Science also applies a direct Welch comparison to paired changes. OpenScience (Synthetic Sciences) identifies opposite NELL1 directions, but its report bases the cross-regimen discussion on separate within-regimen significance results. The `EpiGraph` artifacts carry the argument through the direct statistical test and the alternative label branch. This places a general coding agent at the level of a specialized analysis while retaining the exact premises, protein-indexed evidence, and revision history needed to explain the result.

E.9 Immune Clustering: The Direction of a Relation Is a Prerequisite

The task asks which sepsis immune signatures group together when their scores are correlated. A signature compresses a biological program into a numerical score. Two signatures can increase together across samples, or one can increase as the other decreases. That direction matters: immune activation and immune suppression can be strongly related while describing opposite states.

The completion pass adds the missing sensitivity helper and the script run command to the report. Final inspections check the saved membership and correlation tables and confirm that both required deliverables are nonempty.

To turn the question into a numerical relation, the recorded execution identifies 27 continuous scores with no missing values across 3,948 sample accessions. It uses Spearman correlation (Spearman, 1904), which compares rank order and therefore captures monotonic relationships without requiring a linear scale. A correlation ρ near $+1$ means that the ranks move together; a value near -1 means that they move in opposite directions. The distance $d = 1 - \rho$ preserves this distinction. In the saved output, the lymphoid protective and lymphoid scores have $\rho = -1$, giving $d = 2$. Taking an absolute value would instead give $1 - |\rho| = 0$, placing these oppositely oriented scores at zero distance. This arithmetic explains why the sign belongs in the analysis contract before clustering begins.

To construct interpretable groups, average-linkage clustering joins groups using their average pairwise distance. The five-group summary has sizes 6, 3, 3, 2, and 13, accounting for all 27 scores. The six-score adaptive/lymphoid group has mean within-group $\rho = 0.778$; its members include adaptive, protective, and lymphoid-protective scores. The 13-score dysregulation group has mean $\rho = 0.618$ and includes inflammopathic, detrimental, and myeloid scores. Figure 23 shows each group's mean and observed minimum–maximum correlation alongside reported signed pairs. These ranges describe the pairs inside a group. The trace also records alternative cluster counts: the silhouette score, a measure of separation (Rousseeuw, 1987), is 0.540 at two groups and 0.431 at five. The five-group view supplies a more detailed biological description; the separation diagnostic remains visible.

A Execution meta-graph

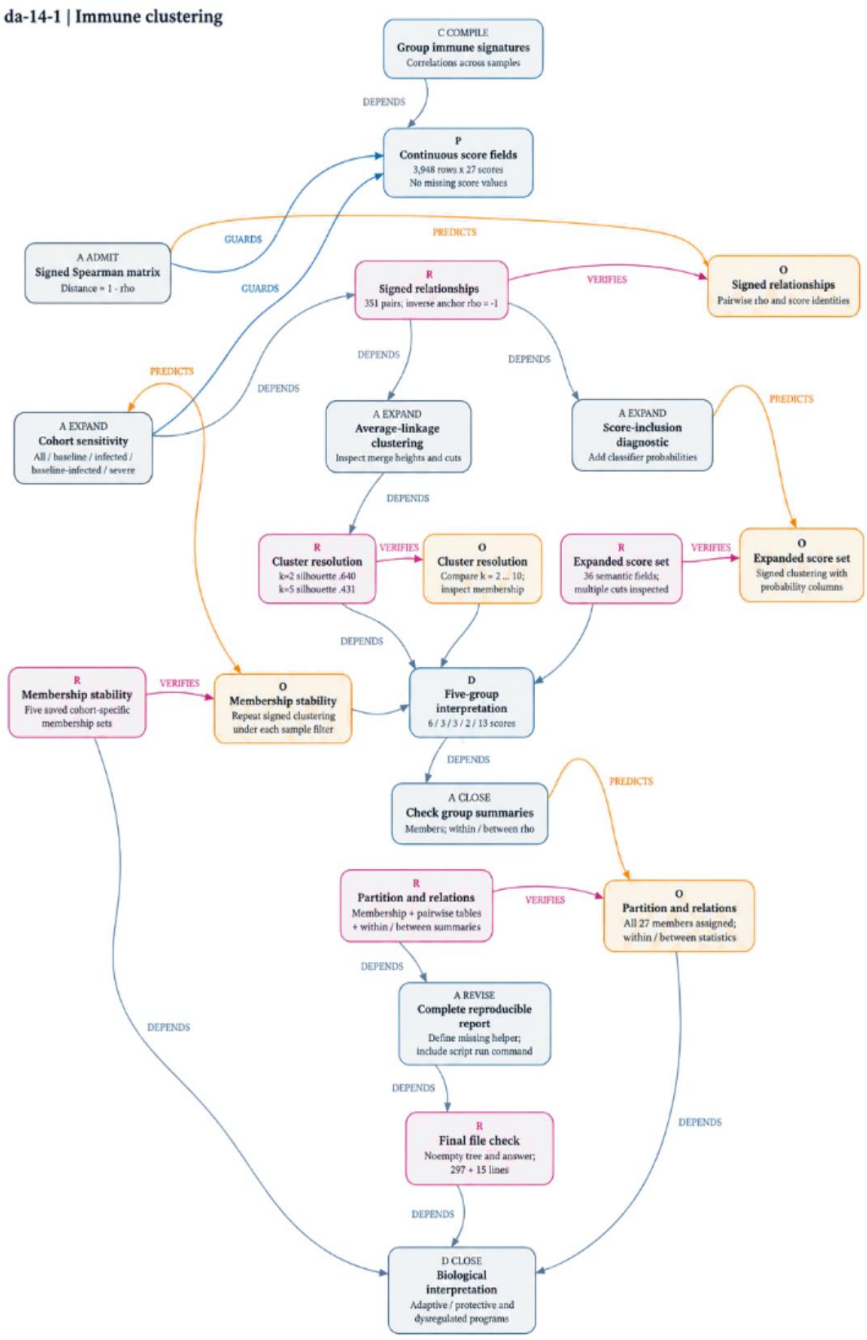


Figure 22: Execution meta-graph for da-14-1. Signed Spearman correlations support cluster-cut selection, cohort sensitivity, and a probability-inclusive diagnostic. Membership, signed relations, and group summaries supply distinct receipts for the biological interpretation.

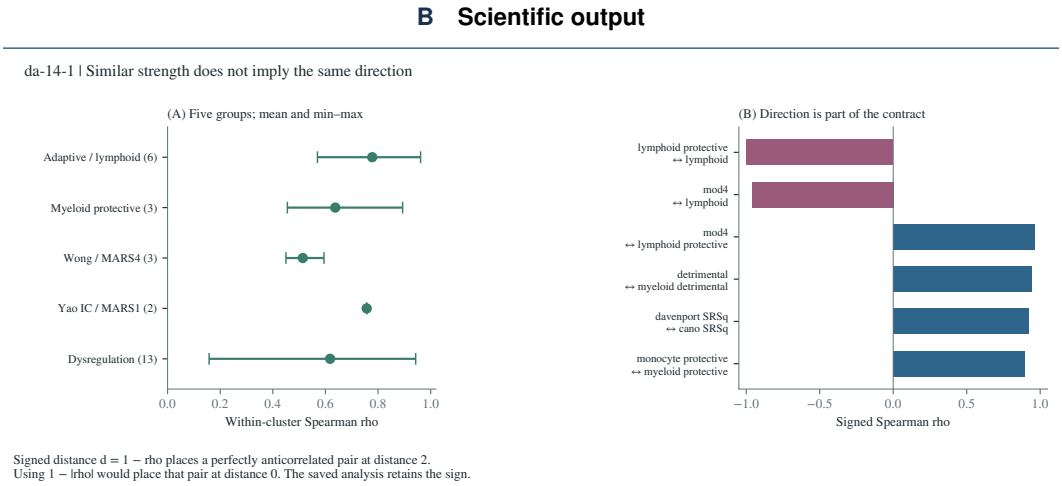


Figure 23: The signed relation connects the immune-signature graph to its scientific output. The left panel summarizes correlations within five groups; the right panel retains the positive or negative direction of six recorded signature pairs, including the $\rho = -1$ pair.

To locate the gain, Biomni scores 83: its numerical correlation and clustering receive full credit, while the rubric identifies incomplete cluster interpretation (+10 available points) and biological synthesis (+7). Codex scores 90: it gives a coherent biological interpretation but uses Pearson correlation for its primary result, with Spearman as a sensitivity analysis. The ten-point difference is entirely in the correlation criterion. `EpiGraph` combines a primary rank-based relation, explicit membership, numerical anchor pairs, and an explanation of adaptive/protective versus dysregulated programs, receiving 100.

To connect these decisions to the graph, `Compile` turns "which signatures group together" into a relation and a reporting contract. The signed relation `Guards` the clustering action. The action `Predicts` an assignment for every score, and the membership table `Verifies` the complete partition. Biological interpretation remains a separate deliverable connected to these groups. Thus the edge carries a concrete scientific choice: changing the definition of similarity changes what a cluster means.

Wisp Science supplies a concrete cross-system contrast. Its analysis uses the unsigned Pearson distance $1-|r|$ and reports $r = -0.83$ between adaptive and inflammopathic programs. That coefficient gives distance 0.17 under the unsigned rule and 1.83 under the signed rule (Figure 29c). The former emphasizes strong coupling; the latter preserves their opposing direction. The `EpiGraph` graph makes this semantic choice a premise of clustering and verifies it with measured signed anchor pairs. OpenScience (Synthetic Sciences) also uses primary Spearman correlation, while EvoScientist and Open Science (AI4S Research) use primary Pearson correlation with rank-based checks. The resulting comparison explains what each cluster represents, and why the relation belongs in the graph before a clustering action is admitted.

E.10 Survival Analysis: Verify the Unit Before Testing the Claim

This task asks whether the microbial genus Kocuria is associated with poor prognosis under an explicit rule: a hazard ratio greater than one and a raw p-value below 0.05. A Cox model (Cox, 1972) relates a measured abundance to the rate of an event over follow-up. A hazard ratio above one indicates a higher fitted event rate as abundance increases. The rule supplies a precise predicate for the final answer: $\phi(o) = \mathbb{I}[\mathrm{HR} > 1 \ \wedge \ p_{\mathrm{raw}} < 0.05]$. The requested output is the fitted estimate, analysis conditions, and the yes/no verdict obtained by evaluating ϕ. The analysis unit specifies how repeated measurements from the same person enter the fit.

To establish the analysis unit, `EpiGraph` matches 457 unique participants with 173 observed events and uses the mean abundance within a participant for its primary analysis. Its saved sensitivity computation makes the consequence of this choice visible. First-row, mean, sum, maximum, and

A Execution meta-graph

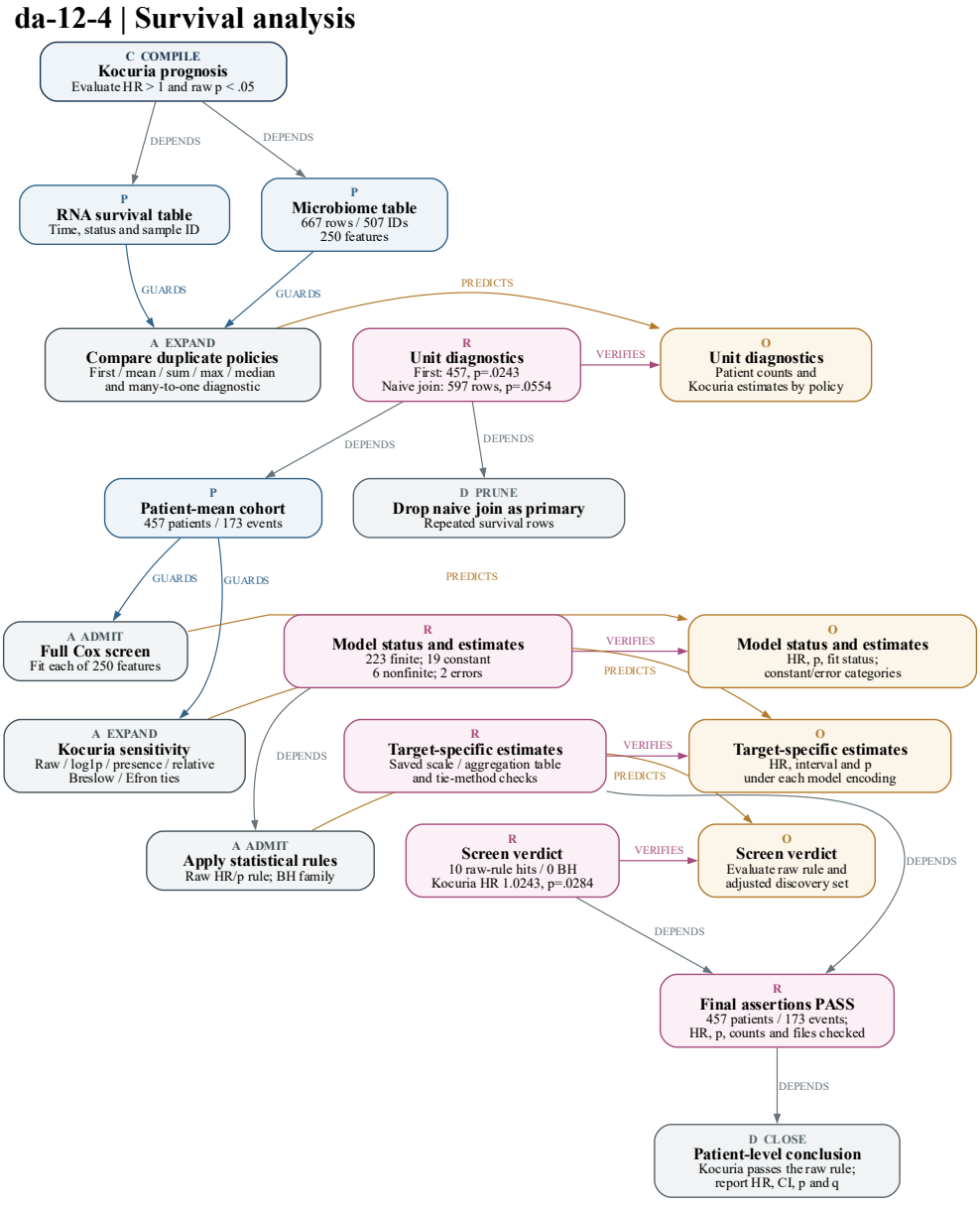


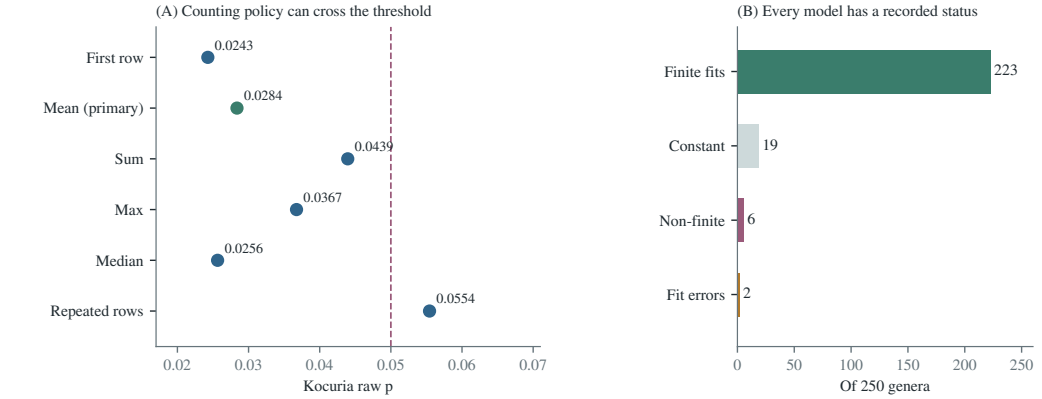
Figure 24: Execution meta-graph for da-12-4. Duplicate-policy diagnostics establish the patient-level unit. The Cox screen produces fit-status receipts before threshold evaluation; target-specific encoding and tie-method checks form a separate branch.

median aggregation give Kocuria p-values of 0.0243, 0.0284, 0.0439, 0.0367, and 0.0256, respectively. Treating repeated rows as separate observations expands the input to 597 rows with 240 event entries and gives $p = 0.0554$. That branch crosses the task's 0.05 threshold. Figure 25 displays these six finite outputs; the raw receipt also records singular or nonfinite results for other policies. The unit of analysis is therefore an upstream dependency of the verdict.

To verify the screen, the agent records a status for each of 250 genera: 223 finite fits, 19 constant features, six nonfinite fits, and two fitting errors. This partition adds to 250 and makes the coverage of the calculation explicit. The primary Kocuria fit gives HR $= 1.0243$, a 95% interval of $[1.0025, 1.0464]$, and raw $p = 0.0284$. The final command returns "summary assertions: PASS." Under the requested rule, the answer is yes. The same receipt reports BH-adjusted $q = 0.8664$; ten genera pass the nominal poor-prognosis rule, and zero pass the corrected rule. The two verdicts answer different predicates, both of which are explicitly available.

B Scientific output

da-12-4 | Fix the analysis unit; apply the declared acceptance rule



Primary: 457 participants, 173 events; Kocuria HR = 1.0243, raw p = 0.0284.
Requested HR > 1 and p < 0.05: satisfied. Across-genus BH q = 0.8664 is reported separately.

Figure 25: The analysis unit is an upstream dependency of a survival verdict. The primary mean aggregation has $p = 0.0284$; treating repeated rows as separate observations gives $p = 0.0554$. The second panel records the statuses of all 250 genus fits.

To explain the improvement, Codex also answers yes and obtains 86 points. Its evaluation attributes the 14-point gap to the Kocuria result: summing duplicates yields a reported $p = 0.0428$, whereas the recorded `EpiGraph` execution documents mean and first-row estimates within the evaluator's expected tolerance. Biomni scores 93, with a seven-point gap in reporting completeness. The three scores and their criterion-level reasons are retained together with the aggregation receipts. The gain concerns a fully specified calculation of the requested result.

To expose the dependency, the matched participant table and aggregation policy `Guards` the modeling action. The expected observation contains a finite estimate, fit status, and the evaluated yes/no verdict. The fit record supplies the estimate; the final assertions check the observed values and affirmative verdict in this execution. `Verifies` connects these receipts to their specific evidence obligations. A change in duplicate policy can be followed through its descendants: input rows, event entries, fitted coefficient, and final threshold comparison. This makes the graph's dependencies directly inspectable in scientific output.

The comparison executions show the same dependency across systems: answers (Figure 29b). EvoScientist averages replicates, retains 457 matched IDs and 173 events, and reports Kocuria $p = 0.0274$, close to the `EpiGraph` primary result. OpenScience (Synthetic Sciences) identifies 457 overlapping IDs but retains 597 joined rows; its reported $p = 0.0517$ falls above the task threshold. OpenScience (AI4S Research) Research) aggregates by summation and reports $p = 0.0428$.

Wisp Science averages replicates but models log-transformed abundance, obtaining $p = 0.517$. Aggregation and predictor scale are thus part of the question answered by the model. The within-execution sensitivity results above show how `EpiGraph` makes these choices explicit before accepting a verdict, while the genus-status partition and final assertions provide executable receipts for the completed screen.

E.11 Cancer Genomics: Carry Every Calculation into the Scientific Answer

The task asks how genomic alterations differ across four breast-cancer receptor subtypes. HR denotes hormone-receptor status; HER2 is a receptor whose status defines another treatment-relevant axis. Their combinations form HR+HER2−, HR+HER2+, HR−HER2+, and triple-negative breast cancer (TNBC). The analysis must connect a clinical subtype to the genomic measurements of the same sequenced sample, then explain the resulting landscape in biological terms.

The primary-receptor label check first normalizes the source vocabulary, then records 1,554 concordant and 164 discordant labels among 1,718 samples with known primary subtype. Analysis groups use the sequenced-sample HR/HER2 fields. The completion pass corrects the reported raw clinical-table width to 35 columns, adds the regeneration command and screen counts, and checks the resulting files and script syntax.

To establish identity and denominators, the saved execution aligns 1,918 clinical samples with the copy-number table and 1,834 with mutation records. It checks the 84 samples without mutation rows against the clinical table and finds recorded tumor mutational burden (TMB) equal to zero in all 84. Subtype assignment uses the sequenced sample's receptor labels. Of 1,718 analyzable samples with a known primary-tumor subtype, 164 disagree with that earlier label. These checks make both the sample identifier and the label source visible before interpretation. Excluding 104 unknown or equivocal assignments leaves 1,814 samples, split into denominators 1,362, 132, 70, and 250.

To make the scientific output readable, Figure 28 reconstructs 40 saved event-frequency cells. Every percentage carries an event count and a subtype denominator. A point mutation changes a gene's sequence; amplification increases its copy number; deep deletion removes copies. These events are kept distinct in the analysis, and combined rows use an explicit union. ERBB2 mutation or amplification occurs in 107/132 (81.1%) HR+HER2+ samples and 64/70 (91.4%) HR−HER2+ samples. RB1 mutation or deep deletion occurs in 42/250 (16.8%) TNBC samples, compared with 28/1,362 (2.1%) HR+HER2− samples. The common denominator makes these subtype patterns interpretable.

To follow a result into the final synthesis, consider TMB, which summarizes the recorded number of mutations per megabase. The trace contains subtype medians, interquartile ranges (IQRs, the middle half of observations), and a Kruskal–Wallis comparison across the four distributions (Kruskal & Wallis, 1952). TNBC has the highest median, 0.167, compared with 0.100 in both HR+ groups; the global test gives $p = 1.82 \times 10^{-9}$. Figure 27 shows the recorded medians and IQRs. These are sample-level summaries, as are the source calculations. The final answer carries the TMB result into the subtype explanation and also identifies ERBB2 point mutations in HER2-negative groups: 53/1,362 (3.9%) HR+HER2− and 13/250 (5.2%) TNBC. This distinction connects a broad receptor class to a more specific genomic subgroup whose biological interpretation is a requested deliverable.

A Execution meta-graph

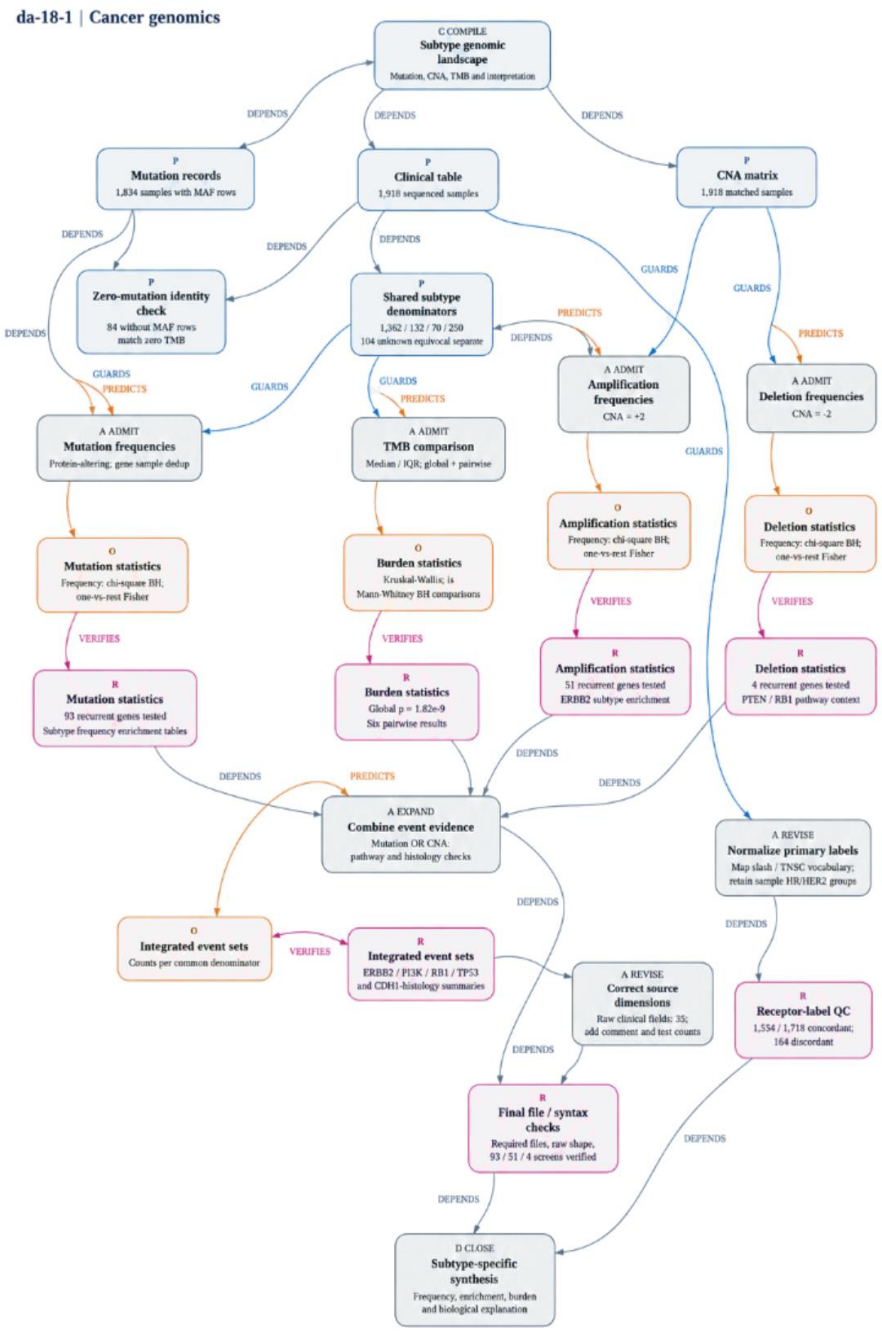


Figure 26: Execution meta-graph for da-18-1. Independent clinical, CNA, and mutation inputs establish shared subtype denominators. Mutation, amplification, deletion, and TMB branches produce separate statistical receipts; combined events connect them to the biological synthesis.

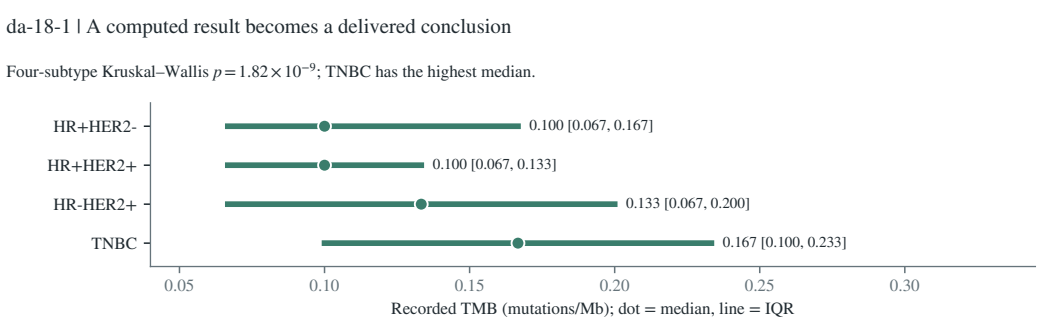


Figure 27: Closing the path from a computed result to the final answer in da-18-1. Points are subtype TMB medians and lines are interquartile ranges, all from the saved execution. The global test and the higher TNBC median also appear in the reported answer. The rubric credits this delivered TMB analysis and its integration into the biological synthesis.

B Scientific output

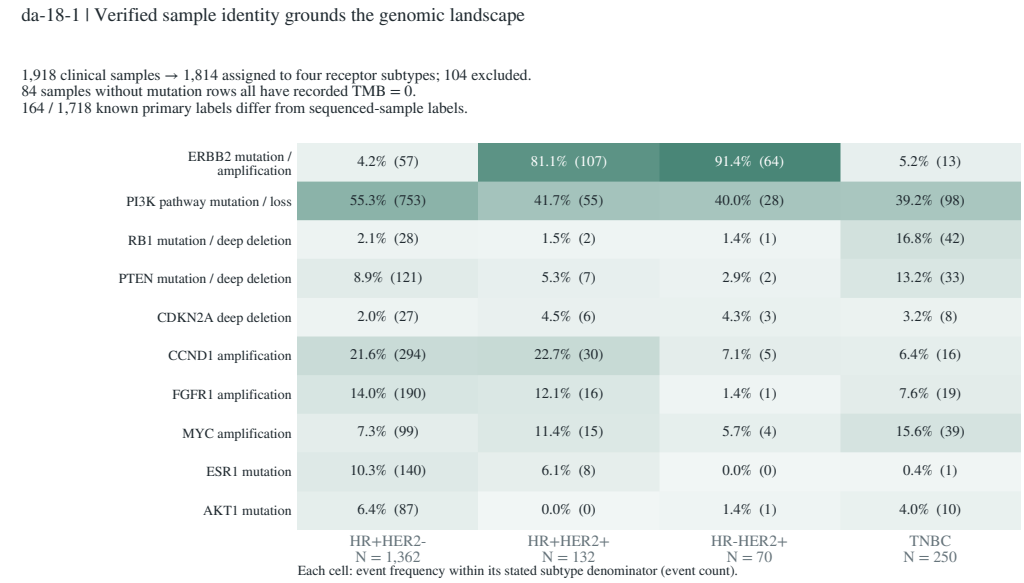


Figure 28: Sample identity, event definitions, and denominators connect the genomic execution graph to an interpretable landscape. Each of the 40 cells shows a saved event count and its percentage within the subtype denominator.

To identify the scored difference, both baselines receive full credit for matching files, assigning subtypes, and separating mutation from amplification. Codex scores 82: the rubric assigns the 18-point gap to TMB reporting (+8) and final synthesis (+10). It had computed a TMB comparison in its trace, but omitted IQRs and the TMB comparison from its final answer. Biomni scores 90, with the remaining ten points tied to interpreting rare ERBB2 mutations in HER2-negative disease. `EpiGraph` closes the numerical and interpretive outputs together and receives 100. The identity checks establish the scientific foundation; the scored advantage is completion of the connected explanation.

To connect the calculation and its delivery, sample identity and subtype labels `Guards` each frequency calculation. The requested report `Predicts` distributions, comparisons, and biological interpretation, while tables and the final synthesis supply different receipts. `Close` requires the path from calculation to delivered conclusion to be complete. A useful intermediate result therefore remains connected to the answer it was computed to support.

The added systems provide useful points of contact with this complete explanation. OpenScience (Synthetic Sciences) reports the same four subtype denominators and a TNBC mean TMB of 0.192 mutations per megabase. Open Science (AI4S Research) reports the TNBC median of 0.167 and the same global $p = 1.82 \times 10^{-9}$. EvoScientist connects subtype mutation and copy-number patterns to biological interpretation. The `EpiGraph` result brings these dimensions together with denominator-bearing event counts, distributional summaries, and rare ERBB2 mutations within HER2-negative disease. Its 100-point answer exceeds the 72–82 points of these three systems and provides a connected route from a sequenced sample to an interpretable subgroup. Here `Close` is a scientific reporting operation: all the quantities needed for the explanation remain attached to their supporting calculations.

E.12 Evidence Obligations as a Route to Scientific Specialization

To understand how a general coding agent acquires scientific competence, we follow the choices that give a result its meaning. The same molecular table can support different comparisons: between people or between repeated measurements, between two treatments or within each treatment, between coordinated immune programs or strongly opposed ones. These choices precede the statistical calculation. In `EpiGraph`, they become premises connected to actions by `Guards` edges; requested outputs become expected observations connected by `Predicts`; and saved calculations provide receipts connected by `Verifies`. The resulting graph exposes which scientific question each number answers.

The extended comparison follows the published analysis records of EvoScientist, OpenScience (Synthetic Sciences), OpenScience (AI4S Research), and Wisp Science alongside the original Codex, Biomni, and `EpiGraph` executions. Screening the 50 task scores identifies cases where improved answers can be examined through concrete intermediate results. We develop six such cases, from cohort construction to molecular interpretation. Figure 29 opens three of the decisions that are particularly revealing across systems; the following walkthroughs connect them to the saved calculations and final biological answers.

Metabolomics gives a clear example of maintaining a scientific question through a sequence of transformations. Codex + `EpiGraph` receives 83 points, compared with 48 for Codex, 63 for Biomni, and 50–68 for the four additional systems. Its execution keeps the potato-versus-grape comparison attached to participant-level measurements, a 435-feature eligible set, and 16 primary mediation tests. The comparison records reveal why those dependencies matter: one analysis substitutes rice for the unavailable grape group under its classification, while another enters replicate columns as independent observations. The graph's advantage is concrete: the identity of the comparison and the unit of inference remain visible all the way to the biochemical conclusion.

Survival and immune clustering expose two further meanings of a prerequisite. The survival model depends on how repeated measurements are combined and on whether abundance is modeled on its original or logarithmic scale (Figure 29b). The clustering depends on whether similarity means a shared direction or a strong relation of either sign (panel c). In both cases, `EpiGraph` binds the numerical result to the choice that produced it. Its saved assertions then check a precise object: the survival predicate, the complete cluster membership, or the signed correlation used to construct a distance.

Paired proteomics shows the depth of the resulting scientific capability. Codex + `EpiGraph` reaches 100 points, matching EvoScientist and both Open Science systems, while preserving within-regimen tests, a direct comparison of changes, and a sensitivity branch for disputed time labels. The direct comparison identifies a corrected NELL1 difference even when the pooled mean change is almost zero. The same six graph operators organize this protein-level reasoning and the cohort-level reasoning above. Scientific knowledge determines the contents of the nodes; the engineering of dependencies determines how those contents license an action and close an answer.

E.13 What the Executions Actually Produce

The execution graphs have tangible scientific outputs. Across the six cases, we retain 54 original files, including 31 CSV or TSV tables, analysis programs, machine-readable summaries, and final reports. A CSV or TSV stores one observation or result per row; a JSON summary gives named

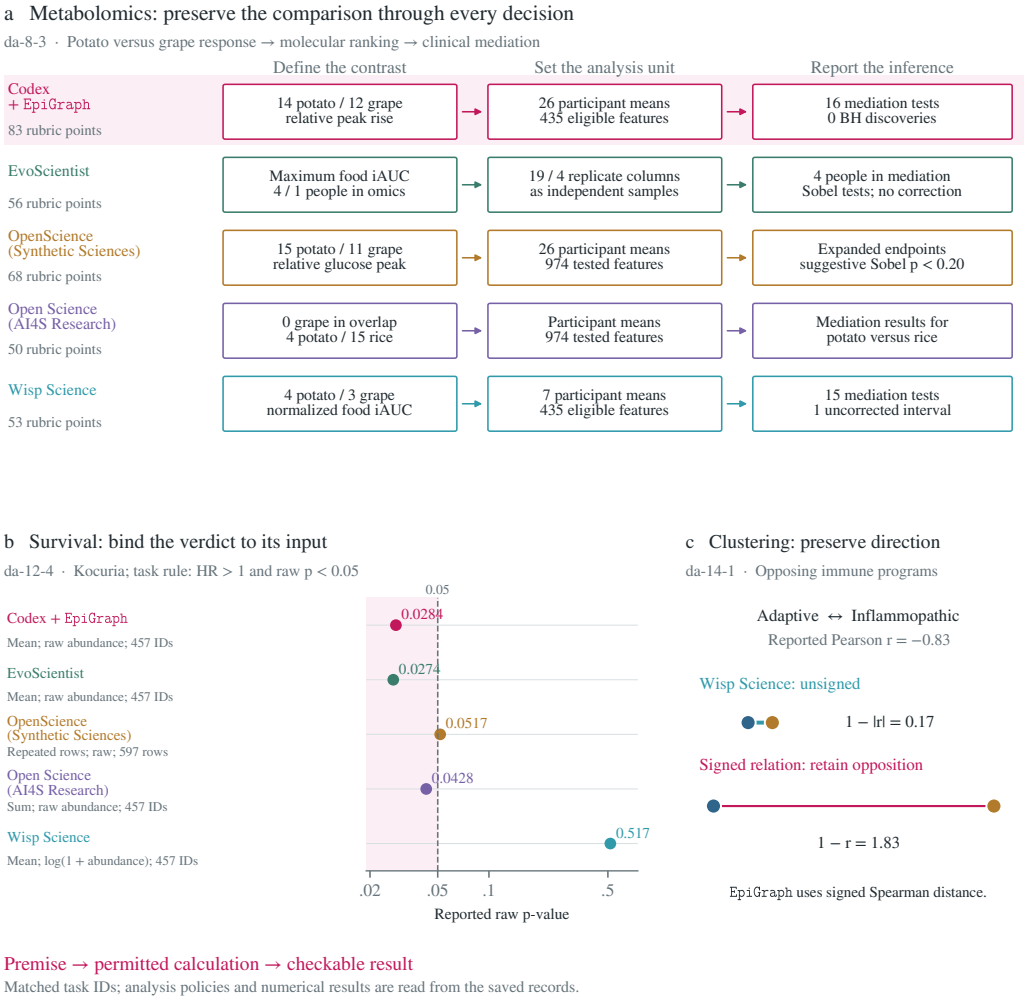


Figure 29: Scientific decisions exposed by matched execution records. (a) On metabolomics task da-8-3, the five paths show the reported contrast, analysis unit, and mediation inference; rubric scores are from the Gemini evaluation. iAUC denotes incremental area under the glucose curve, and BH denotes Benjamini–Hochberg correction. (b) On survival task da-12-4, the reported Kocuria p-values are displayed together with the aggregation and predictor scale used by each system. Shading marks $p < 0.05$; all five fitted hazard ratios exceed one. (c) Applying signed and unsigned distance formulas to the same Pearson coefficient reported by Wisp Science makes the effect of discarding direction explicit. The `EpiGraph` analysis uses signed Spearman distance; its measured rank-correlation examples are developed in Section E.9.

quantities that another program can read directly. Together, these files let a reader move from a result to its calculation and acceptance check. Figure 30 presents a small window into each case. The files come from the corresponding execution directories and preserve the event streams and reports used in the figures. The displays below typeset their contents and visualize their saved values.

The immune-subtyping program saves the adopted threshold as 1.65 and implements the four subgroup definitions. Its execution receipt checks the rule against the supplied labels and reports zero mismatches before presenting the patient-level counts. This makes the classification procedure inspectable alongside the 1,303-patient denominator. In metabolomics, the top-ten table keeps feature identity, biochemical annotation, effect size, sample counts, raw p, and adjusted q together. For example, the first-ranked feature has $p = 0.00164$ and $q = 0.28869$. The file simultaneously delivers the requested ranking and the information needed to interpret its statistical strength. The separately



Figure 30: Original deliverables from six `EpiGraph` executions. Each card names a saved artifact and displays source text, table entries, or an associated execution receipt. A links an executable threshold to its label check; B retains ranking and corrected statistics together; C keeps two label interpretations in separate result tables; D preserves relations and cluster membership; E binds the survival result to a machine-readable population; F carries subtype statistics into the report. Table entries are rounded for display; the exported original files preserve full precision.

saved mediation table records the endpoint, indirect effect, bootstrap interval, and correction family for each candidate.

In genomics, 13 saved tables connect subtype denominators to mutation, amplification, deletion, and tumor-mutation-burden results. The subtype summary distinguishes samples from patients; the TMB table carries a sample count, median, quartiles, mean, and zero count for each subtype. The final answer uses the saved TNBC median of 0.167 mutations per megabase and reports the global test. This is a concrete calculation-to-delivery link: a result produced by the analysis appears with its biological interpretation in the answer. The Codex baseline evaluation locates an eight-point gap in TMB delivery and a further ten-point gap in synthesis, while `EpiGraph` receives 100 points. We next inspect three artifact chains in greater detail.

Proteomics: a premise has an executable consequence. Figure 31 follows the saved program, two result tables, and the command that reads them back. The program derives subject identifiers from sample-name prefixes and intersects them across baseline and six months. It subtracts measurements aligned by those identifiers under the metadata-defined time labels. This implements the declared `Guards` condition for the paired analysis.

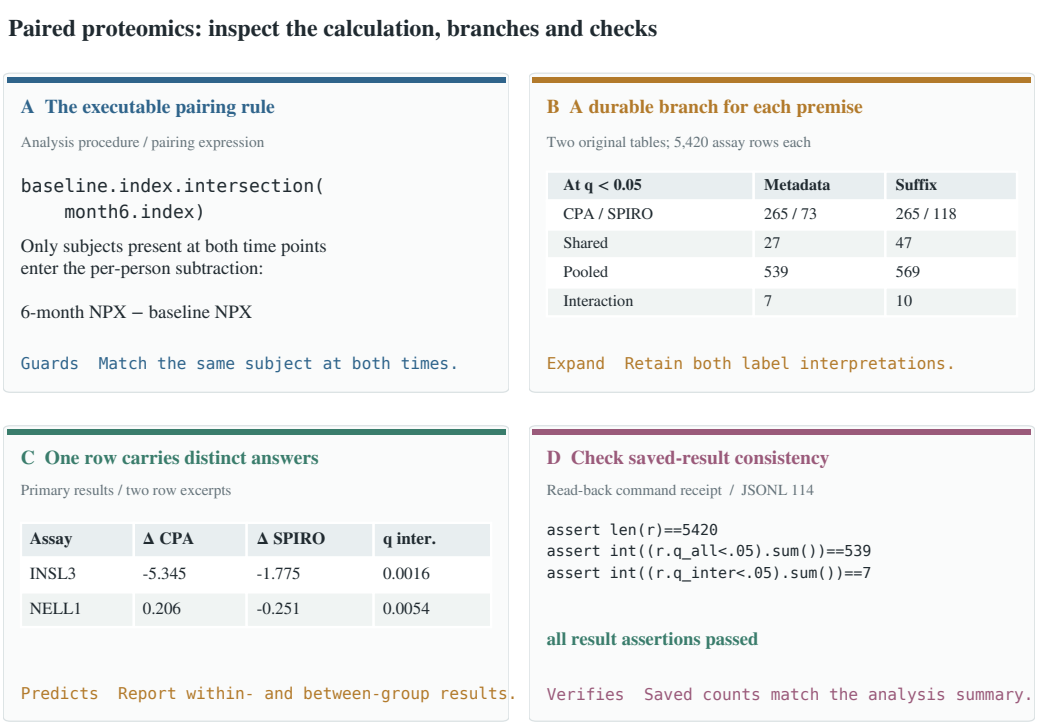


Figure 31: Inspecting the persisted proteomics artifacts for da-13-1. A shows the original program's same-subject intersection, line-wrapped. B summarizes corrected discoveries from the two saved 5,420-row tables. C displays two rows from the primary table; changes are six-month minus baseline NPX, and the interaction tests the difference between regimen changes. D quotes the original read-back command and its output (JSONL 114): the assertions check table length and agreement of saved discovery counts with the analysis summary. Panels A–C show the pairing procedure, label sensitivity, and saved statistical results; panel D checks artifact consistency.

Each result table contains 5,420 assay rows. A row carries the number of complete pairs, mean change, and raw and corrected tests for each regimen, the pooled analysis, and the interaction comparison. The scientific consequences of these separate columns are visualized in Figure 21: NELL1 has opposite regimen means and a near-zero pooled change, while the time-label sensitivity affects SPIRO and its downstream comparisons. Keeping the full tables from both interpretations exposes how the reported effects depend on the time-label rule.

Finally, the agent reopens the saved tables and checks the primary table's row count, discovery counts, and shared-set counts against the summaries from the analysis, then checks that the named proteins appear in the final answer. The assertions for 539 pooled and seven interaction discoveries count rows with saved $q < 0.05$; their acceptance predicate is agreement with the reported discovery counts. The command returns "all result assertions passed." This `Close` receipt confirms the primary table length and summary consistency. The pairing program, statistical outputs, and time-label sensitivity branch supply the corresponding records of the analysis and its dependence on the declared design. The final shared/unique comparison closes the five-point reporting gap relative to Codex's 95-point answer; Biomni scores 60 and `EpiGraph` scores 100.

Survival: a result becomes a machine-checkable receipt. Figure 32 shows the original JSON summary, the Kocuria row of the 250-row model table, the executed assertions, and the final answer. The summary records both the 597 rows from a naive match and the 457 distinct participants used for the primary fit. It also records 173 events. Those quantities establish what one observation means before the hazard ratio is interpreted.

The Kocuria row contains its cohort size, event count, fitted coefficient, hazard ratio, standard error, raw p, confidence interval, fit status, and adjusted q. The status tells a downstream reader that this numerical result came from a finite fit. The other model rows retain their own statuses, so the 250

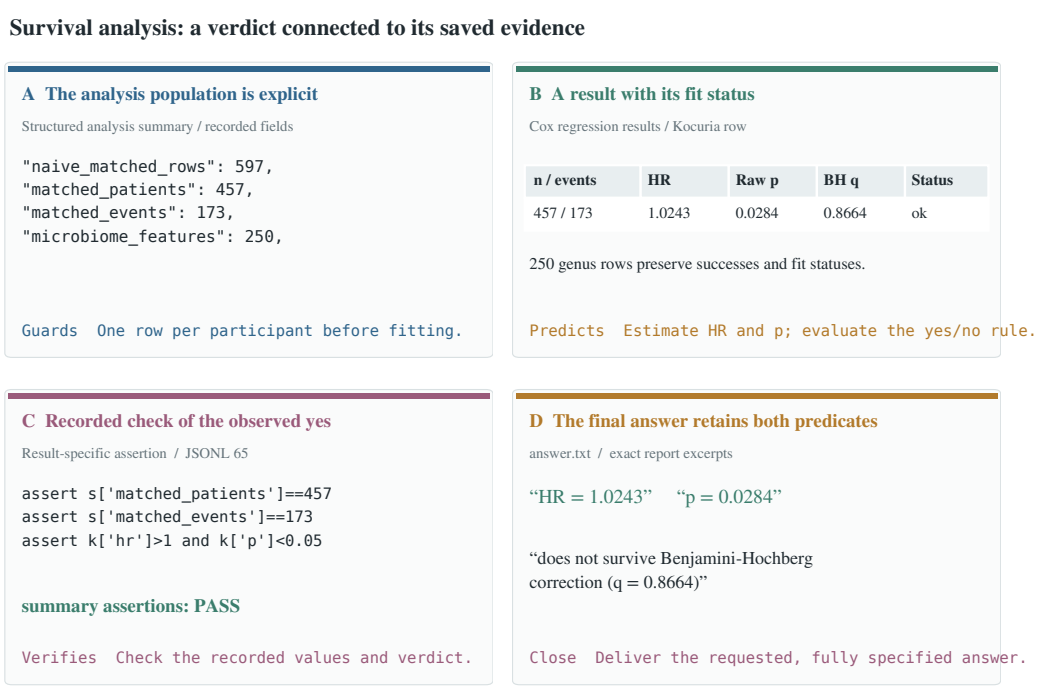


Figure 32: Scientific verdict and execution checks for da-12-4. A records the analysis population; B supplies the fitted Kocuria statistics used to evaluate the yes/no rule; C quotes the original command at JSONL 65, including its assertion of the affirmative result observed in this execution; D reports the raw-rule verdict and multiple-testing result. `Predicts` specifies the estimate and rule evaluation; `Close` checks agreement of the reported verdict with the evidence. Code and report snippets come from the recorded execution; displayed statistics are rounded.

tested genera can be reconciled with the summary's 223 finite fits. The separate sensitivity file preserves the results under alternative duplicate-handling policies.

The task's decision rule is evaluated on the fitted HR and raw p; its output is a yes/no verdict. In this execution, HR $= 1.0243$ and $p = 0.0284$ give "yes," while the corrected rule gives "no" with $q = 0.8664$. The recorded checking command reads the JSON and CSV, checks cohort counts and exact fitted values, and asserts the observed affirmative raw-rule result. Figure 32C quotes this result-specific check. The completed answer reports both verdicts with their estimates and analysis conditions. Here `Close` accepts a report whose verdicts agree with the saved evidence. The 100-point result, compared with Biomni's 93 and Codex's 86, accompanies this fully specified answer.

Immune clustering: the intermediate matrix exposes the relation. Figure 33 visualizes three saved files: the 27-by-27 signed Spearman matrix, the five-cluster membership list, and the merge sequence. The final merge record supplies the plotted order. Every cell is a saved correlation, and every cluster outline follows the saved membership. Reading across one row shows how a signature relates to all the others; the diagonal compares each signature with itself and equals one. Blue cells indicate positive association, purple cells negative association, and near-white cells weak association.

The first cluster contains six signatures, including lymphoid protection, whereas the fifth contains 13 signatures, including the lymphoid score. The gold-marked cells connecting those two named scores equal -1: the rankings move in opposite directions. The saved analysis uses distance $1 - \rho$, so this pair has distance two. Taking an absolute correlation would instead give distance zero for this pair. The matrix makes the consequence of the chosen relation visible across the whole analysis: positive within-group blocks coexist with negative links between groups.

This artifact supplies a concrete view of graph engineering above a task's numerical procedure. The meaning of similarity is a prerequisite; the clustering calculation is an action; a membership list and signed within- and between-group relations are expected outputs. The saved matrix, membership,

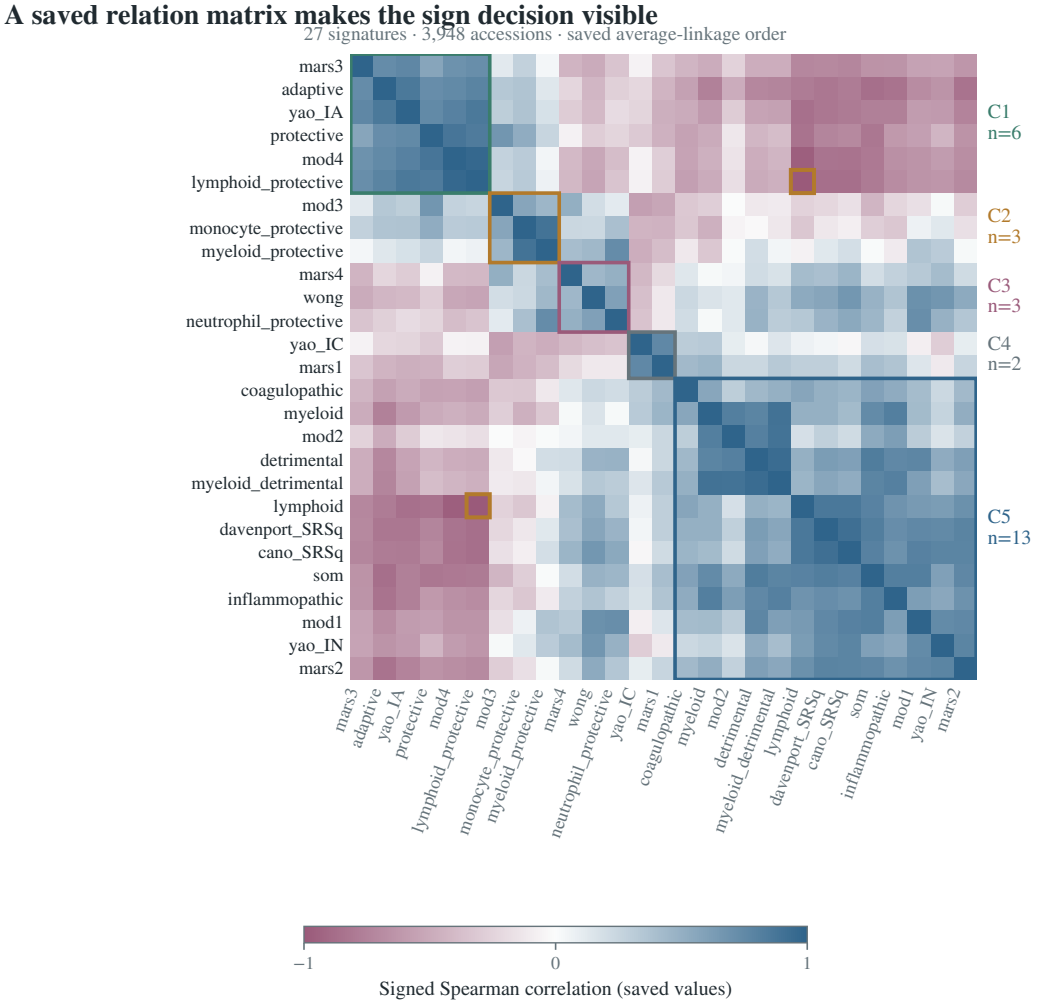


Figure 33: The actual saved correlation matrix from da-14-1, displayed in the order recorded by the recorded execution. The 27 signatures are grouped into clusters of 6, 3, 3, 2, and 13. Colored diagonal outlines use the saved cluster assignments; gold cells identify the perfectly negative relationship between lymphoid protection and the lymphoid score. The original score-name suffix is omitted from axis labels for readability. Matrix values, membership, and merge order are read from persisted files without recomputing correlations or clustering.

and merge table allow those objects to be inspected together. The Codex baseline run uses Pearson correlation for its primary analysis and Spearman for sensitivity, receiving 90 points; the `EpiGraph` run implements the requested Spearman relation as primary and receives 100. Biomni receives 83, with its remaining gap in biological interpretation. The figure lets the reader see the scientific object behind that criterion.

E.14 Why These Graphs Work Across Scientific Tasks

The six cases instantiate distinct dependency structures. Immune subtyping joins cutoff calibration with patient-level aggregation; metabolomics joins phenotype, participant, feature, and endpoint checks; proteomics shares an assay matrix across four test families and two sensitivity branches. Clustering fans out into resolution and cohort diagnostics, survival analysis separates model-status

checks from target sensitivity, and cancer genomics combines mutation, CNA, and burden results under common denominators.

The operators retain their meanings across these structures. `Compile` specifies the requested outputs, `Expand` adds diagnostic branches, `Prune` removes an unsuitable continuation, and `Admit` connects an analysis to its established prerequisites. `Close` checks each result against its evidence obligation. `Revise` updates a contradicted premise, as in the corrected immune cutoff and metabolomics participant join. Shared prerequisites support several actions, and separate receipts close the corresponding expectations.

The complete biomedical comparison is summarized by the task-level score distribution in the main text. The six walkthroughs instantiate the execution-level pattern across distinct scientific tasks.

To identify the execution-level pattern, the graph keeps three decisions separate: what population or object is being analyzed, what evidence would answer the question, and whether the executed calculation supplied that evidence. In immune subtyping the decisive check is the score-to-label rule; in metabolomics it is the connection between ranking and corrected inference; in proteomics it is closure of both set-based reporting and direct comparison. Clustering makes the meaning of a relation explicit, survival analysis binds a verdict to its unit and predicate, and genomics connects computed statistics to the final biological explanation. The advantage can therefore appear as a corrected premise, a `Revise` update, or completion of a nearly finished answer. This separation gives the case traces concrete form: legality is carried by verified dependencies, the operators retain their meaning across tasks, and scientific receipts provide checks that executable code can evaluate.

The extended execution comparison makes the role of those dependencies visible across systems. A substituted food group changes the metabolomic contrast; duplicated survival rows change the fitted verdict; unsigned correlation joins opposing immune programs; and separate within-regimen significance leaves a direct protein comparison open. Figure 29 and the protein-level results locate these choices at the corresponding prerequisites and expected observations. The same representation can therefore encode a population, a statistical unit, a relation, or a test family while the six operators retain their semantics.

The extended system comparison also connects this mechanism to specialization. Codex + `EpiGraph` reaches 67.48 mean rubric points and attains full-credit results on cases also solved by dedicated scientific systems (Figure 11 and Figure 4). The saved executions expose the capability behind those outcomes: biological choices become explicit objects whose consequences can be followed through the calculation. The NELL1 example preserves a regimen-associated pattern through separate pooled and interaction tests; the immune example checks how labels are constructed; the survival example fixes the unit of inference before interpreting the fitted effect. The graph is engineered above these scientific tasks, while the evidence that satisfies each edge is specific to the scientific question.

F RQ3: Module Ablations and Failure Inventory

The module supplement isolates the graph representation and execution operators. It reports the ablation protocol and failure inventory, complementing outcome results with mechanism-focused comparisons. We examine the two additional ablations, matched-nograph and no-verifies, using all 198 GPQA-Diamond and 100 BeyondAIME tasks for GPT 5.6 Terra and Claude Sonnet 5. The Full Graph results remain the original main-paper results. The table reports matched outcomes for the two ablation variants. A wrong answer can arise from a reasoning error, a numerical error, or final-answer extraction; an absent event marks a missing structured record, whose cause is classified separately from the answer outcome.

Checklist construction. The matched-nograph condition is a checklist execution policy. Its checklist names the task goal, scope boundary, prerequisites, expected results, evidence to record, deliverables, and the checks to run after an error. These entries follow the plan-then-execute and reason–act styles used in prior agent policies (Wang et al., 2023b; Yao et al., 2023b). The graph condition carries the same task-facing requirements as explicit node classes and connects them with `Guards`, `Predicts`, and `Verifies` relations. Thus the comparison keeps the execution requirements aligned while changing their representation from an ordered checklist to a dependency graph.

Table 7: Failure inventory for the two newly run ablations. Counts are computed within each ablation condition; Full Graph is reported separately in the main results. "Wrong" denotes an incorrect extracted answer, "state" an abnormal run status, and "no event" a cell without a structured execution event.

Benchmark	Model	Condition	Wrong	State	No event
GPQA-Diamond	GPT 5.6 Terra	matched-nograph	29/198	0	2
GPQA-Diamond	GPT 5.6 Terra	no-verifies	35/198	0	7
GPQA-Diamond	Claude Sonnet 5	matched-nograph	44/198	1	8
GPQA-Diamond	Claude Sonnet 5	no-verifies	49/198	0	7
BeyondAIME	GPT 5.6 Terra	matched-nograph	32/100	0	7
BeyondAIME	GPT 5.6 Terra	no-verifies	30/100	0	10
BeyondAIME	Claude Sonnet 5	matched-nograph	45/100	1	7
BeyondAIME	Claude Sonnet 5	no-verifies	46/100	4	16

The two conditions share many wrong tasks: 20 for Terra and 33 for Claude on GPQA-Diamond, and 21 and 31 on BeyondAIME. The failure inventory above records the condition-level counts for wrong answers, abnormal states, and missing events.

The overlap matters for interpreting the table. Removing the graph does not create an entirely new set of difficult questions; it leaves a substantial core of failures shared with the graph condition. The additional failures and missing events therefore describe execution organization rather than assigning every error to one edge or one operator. The pattern is also model- and benchmark-dependent: Terra's no-verifies condition has more wrong GPQA answers and more missing events than matched-nograph, whereas BeyondAIME changes the balance of wrong answers while increasing missing events. Claude shows the clearest recording fragility on BeyondAIME, where no-verifies has 16 no-event cells and four abnormal states.

These counts support a mechanism-level conclusion. A checklist can preserve the names of prerequisites and deliverables, but it does not preserve the support relations that tell the agent which action is currently licensed or which receipt is still required. Removing those relations can leave a final answer wrong, or leave the execution without a structured event from which the answer could be audited. The failure inventory therefore complements the main accuracy ablation by linking the negative outcome to two observable pathways: incorrect answer production and loss of the execution record needed for closure.

G RQ4: Prospective Intervention Consequences

The intervention supplement tests how changes to premises, receipts, and labels alter the subsequent execution trace and judged result. It is kept separate from the retrospective biomedical cases above. We score the original `da-14-3` trace and the three valid prospective intervention traces with the same Gemini 3.1 Pro BiomniBench-DA rubric used for the benchmark results. Each trace is judged independently from its saved `trace.md` and `answer.txt`. The intervention study therefore gives a case-level performance readout in addition to the fork/commit trace analysis.

The case is reported on the six global process dimensions under the paper's equal-criterion index: data handling (C1–C2), method selection (C3), statistical rigor (C4), biological interpretation, scientific reasoning (C5), and source reliability (C6). This task has no biological-interpretation criterion, so that dimension is recorded as 0 for every trace. Criteria C1–C5 carry positive point weight; source reliability is recorded at zero weight for this task. The corresponding dimension indices are:

Trace	**Data handling**	**Method selection**	**Statistical rigor**	**Biological interpretation**	**Scientific reasoning**	**Source reliability***
Original trace	100	100	100	0	100	100
Premise mutation	100	50	100	0	50	50
Receipt deletion	100	50	100	0	100	100
Label conflict	100	100	100	0	100	100

*The biological-interpretation dimension is explicitly 0 because this task has no criterion mapped to it. The source-reliability dimension is shown because it is part of the global process profile; its task-specific maximum is zero. The dimension index uses the same A=100, B=50, C/missing=0 mapping as the main-text process profile.

Table 8: Gemini 3.1 Pro rubric scores for the original `da-14-3` trace and the three valid prospective interventions. Criterion columns show the rubric level selected by the frozen BiomniBench judge; C6 is recorded by the rubric but has zero point weight for this task.

Trace	Score	Δ	C1	C2	C3	C4	C5	C6
Original trace	100	0	A	A	A	A	A	A
Premise mutation	66	−34	A	A	B	A	B	B
Receipt deletion	85	−15	A	A	B	A	A	A
Label conflict	100	0	A	A	A	A	A	A

The performance profiles in Figure 7 summarize the original trace and the three replay variants. The compact table below records the directly auditable counts for each fork.

Table 9: Auditable counts for the three intervention replays.

Variant	Fork	Graph nodes	Records affected	Result
Premise	`cohort`	12	162 rows / 156 patients	1,158 rows / 1,147 patients
Receipt	`cutr`	10	0 rows	0 mismatch
Label	`stanford_96`	8	1 row / 1 patient	counts unchanged

Premise mutation. The original cohort predicate is `baseline & severe`; the intervention adds `condition != non-infected`. The replay removes non-infected records and recomputes the cohort, patient denominator, subgroup counts, proportions, and patient-level sensitivity analysis. The independent score-to-label branch is retained and reports zero mismatches. Data handling and quantitative reporting remain complete, while the changed cohort receives B levels for C3 and C5 (and the corresponding C6 record) because the explicit z-score threshold and its rationale are not carried into the revised analysis. The resulting score is 66, with data handling and statistical rigor at 100. The result shows how a premise change propagates through its descendants and changes the connected rubric obligations.

Receipt deletion. Deleting `cutr` removes the saved label-agreement evidence while leaving the original cohort, unique-patient denominator, duplicate-collapse rule, and z-score rule available. The replay reopens the verification branch, regenerates the high/low labels from the saved z-scores, and checks both axes before allowing patient aggregation to close. The check returns zero mismatches, so the final counts match the original trace. The rubric assigns 85 because the recorded analysis lacks the full threshold rationale and negative-score handling. The replay separates numerical agreement from evidence closure: the receipt is reconstructed and the answer remains unchanged, while the verification record carries less rubric credit.

Label conflict. The third intervention changes one row-level label for `stanford_96`. The graph identifies the patient-level label-consistency and collapse descendants, while the global threshold, cohort, denominator, and all other patients remain committed. The conflict count changes from four to three, and the affected patient moves from mean-score fallback to consensus labeling; that patient is still classified as system-wide. The aggregate counts therefore remain unchanged and all six rubric criteria remain A. The `Revise` update remains local while unrelated work stays committed, and the repaired branch reaches a semantically equivalent final aggregation.

Why the performance pattern matters. The four rubric scores form a structured response. A premise change has the widest dependency footprint and the largest score decrease; receipt deletion preserves the numerical answer while changing the evidence and method record; and a local contradiction is repaired with the final answer unchanged. The six-dimensional profile makes the same distinction visible: premise mutation lowers method selection and scientific reasoning, receipt deletion lowers method selection alone, and label conflict leaves the profile unchanged. The graph connects each scientific-state change to the rubric obligations that depend on it.

Connection to the claims. The receipt-deletion replay shows legality carried by the `Verifies` edge: a downstream aggregation closes after its receipt has been regenerated and checked. The three variants use the same graph-level propagation and the `Revise` operator for a premise, an evidence object, and a label, while the scientific setting makes the consequences inspectable through saved tables, z-scores, label-consistency checks, patient counts, and rubric criteria.

These case-level mechanism results complement the aggregate benchmark results. The benchmark establishes overall transfer and performance, while the intervention study shows how execution changes and how `EpiGraph` preserves or revises scientific work after a targeted disturbance.

H FULL EXECUTION POLICY

The following policy listing is the reproducibility artifact used to instantiate the execution procedure described in the method and evaluated above. `EpiGraph` **execution policy.**

This appendix presents the `EpiGraph` execution policy used in our experiments. Dark headers mark policy modules, teal boxes mark non-negotiable interfaces or gates, and monospaced panels contain operational records that an agent may instantiate through the host's task state and event records.

Policy scope **Use when:** a task has multiple dependent decisions, unresolved information, delegated work, side effects, or evidence-based acceptance and a linear plan may drift; also use when scope, boundaries, metrics, verification, or replanning need to be explicit. **Do not use for:** a one-step reversible action with an already explicit check.

Execution Boundary — Working directory

Work from the selected working directory and keep new files, caches, and outputs there. Use the configured credential adapter, small Git commits, the smallest relevant tests, and bounded cancellable commands. Avoid recursive parent-tree scans and full-tree hashes; these are practical resource limits, not launch admission.

Overview — Execution meta-graph policy

`EpiGraph` is a portable planning discipline for tasks where what must be known, what may be changed, and what counts as done are distinct. It turns a task prompt and the observed current state into a bounded contract and a small, mutable execution meta-graph. Each task instantiates four execution node classes: prerequisites, actions, expected observations, and receipts. The three obligation classes carry the task conditions and evidence; action nodes are the transition vertices admitted between them. The three named edges, `Guards`, `Predicts`, and `Verifies` license an action, declare its observable commitment, and close it with evidence. The six operators `Compile`, `Expand`, `Prune`, `Admit`, `Close`, and `Revise` construct and maintain the task graph. At each state the graph is a branching dependency DAG and the trace records operator-labelled graph-state transitions. The operator order follows the current evidence and unresolved obligations; scientific receipts provide the values that close or revise affected branches.

Use the host's lifecycle and interfaces to maintain graph state. The policy defines decision semantics and records concise decisions, observations, and evidence as execution proceeds.

Behavioral Evaluation — Paired intervention

For a realistic check that this policy changes observable behavior, use the paired prompts and evaluation rubric. Compare the baseline and `EpiGraph` under the same host, model, tools, task state, and permissions, then score only user-visible decisions and permitted receipts under the declared evaluation contract.

Interface with a Task Prompt — Separate input from policy

The **task prompt** is an engineering input for one task. It may provide the objective, context, explicit constraints, permissions, available interfaces, and output requirements. The policy supplies the reusable planning policy.

Do not put a solved decomposition, a preselected action sequence, or a hidden graph into the task prompt when evaluating this policy. If the prompt gives an explicit goal, boundary, or acceptance condition, treat it as a fact and represent any missing decision-relevant field as an uncertainty. An evaluation may keep the task prompt and host fixed while toggling this policy.

Rule 1 — Compile a contract before a non-trivial action

Write the smallest useful contract:

```
C = (Goal, Boundary, Evaluation)

Goal:
  Terminal state, deliverables, claims, and priority relations.

Boundary:
  Allowed scope, authority, paths, data, interfaces, side effects,
  resources, time/cost limits, invariants, and non-goals.
```

```
Evaluation:
  Metric, threshold or rubric, required evidence, aggregation rule,
  independent check, and conditions for failure or uncertainty.
```

If a missing field could change the next decision, acquire it with a bounded information action (inspect, ask, retrieve, compare, or test) before taking a world action. Mark the field unknown while it is unresolved. Do not replace uncertainty with a convenient assumption.

Rule 2 Maintain a small execution meta-graph

Use a graph for facts and relationships that can affect the next decision. Its representation is the host's native transcript, state, or event format.

Node	Meaning
prerequisite	A requirement or observed state that can guard an action.
action	An information or world action proposed for execution.
expected observation	The observable property declared before an action.
receipt	Persisted evidence that can discharge an expected observation.
uncertainty	An unresolved field whose value can change a decision.
decision	Continue, prune, retry, revise, escalate, stop, or complete.

Useful edge meanings include `REQUIRES`, `INFORMS`, `ENABLES`, `Guards`, `Predicts`, `Verifies`, and `REVISES`. The core graph vocabulary is fixed: prerequisites guard actions, actions predict expected observations, and receipts verify those observations. For each guarded action, represent one or more support sets: members of a set are joint prerequisites, while distinct sets are alternative compliant continuations. An action is edge-ready when at least one support set is satisfied, its expected observation is declared, and its receipt obligation is specified. Prune nodes that no longer affect a decision.

Keep **information actions** distinct from **world actions**. Resolving a decision-relevant unknown can be more necessary than executing a short edit; resolving an unknown that cannot change the answer is wasted computation.

Rule 3 Expand the graph with four strategies

The `Expand` operator uses four strategies: backward reasoning, forward reasoning, dialectical comparison, and expectation setting. Choose the strategies that the current uncertainty, risk, and cost justify. They guide dependency expansion within the six-operator execution policy.

1. **Think backward from the question.** Expand what the request really asks, what would count as an answer, which prerequisites and evidence are needed, and which likely interpretation would be a category error.
2. **Think forward from the current state.** Enumerate reachable state changes, candidate actions, dependencies, resources, side effects, and likely observations.
3. **Confront alternatives dialectically.** Challenge assumptions, compare cheaper or safer paths, search for conflicts and counterexamples, and ask what observation would distinguish apparently valid alternatives.
4. **Set an expectation before acting.** Attach a predicted observation or state change, confirming evidence, an unacceptable outcome, and the resulting continue/replan/stop implication to each non-trivial action edge.

After acting, compare reality with the expectation. A surprise is a graph update: preserve the receipt, invalidate descendants whose support no longer holds, and compile a fresh local contract. Continue any descendant that still has an independent satisfied support.

Rule 4 Gate candidate actions with an omission test

Before selecting a non-trivial action, record enough to answer:

```
supports:       Which Goal requirement, Boundary guard, uncertainty,
                or evidence?
alternatives:   Is there a cheaper, safer, or completed substitute?
omission_test:  If removed, does a compliant path remain?
```

```
uncertainty:    Which decision could its observation change?
cost_risk:      Time, tokens, permissions, latency, irreversible effects.
recheck:        What new evidence could change its necessity?
```

Classify the action as:

- **necessary**: omitting it makes a required compliant state or evidence unreachable;
- **diagnostic**: it distinguishes branches that lead to different decisions;
- **optional**: it may improve quality but is not needed for this contract; or
- **redundant**: an equivalent compliant path or evidence already exists.

Use necessary and decision-relevant diagnostic actions as the first ordering class. If compliant continuations remain but no such action is available, admit an edge-ready action on one continuation and choose it by the host's cost, risk, and reversibility policy. An optional action remains admissible when it begins a compliant continuation; it becomes redundant after an equivalent alternative has supplied the same contract or evidence. Defer or prune redundant actions.

Rule 5 Give every delegated action a local contract

When the host uses another worker, pass a bounded local contract rather than a vague request:

```
Task:                 One bounded action.
Goal contribution:    Requirement or state this action supports.
Boundary:             Allowed scope, interfaces, side effects, limits.
Metric:               Pass condition and threshold or rubric.
Evidence required:    Exact receipt, output, diff, test, hash, or log.
Expected observation: What should be true after the action.
On mismatch:          Retry, report, replan, escalate, or stop.
Return:               status | work | observations | evidence |
                      boundary deviations | unresolved questions
```

The worker cannot relax the global Boundary, redefine the Metric, or declare global completion without the required evidence. The host may implement delegation sequentially, in parallel, or asynchronously; this policy does not prescribe a communication protocol. If several actions return concurrently, merge only observations that remain compatible with the current contract and boundary.

Rule 6 Observe, verify, and revise

The graph is a branching dependency DAG at each execution state. The procedure below iterates operator applications as observations arrive; its trace is the resulting sequence of graph states. The `Revise` operator opens an affected branch while retaining descendants with independent support.

```
plan(task, state):
    C <- compile Goal, Boundary, Evaluation
    if a decision-relevant field of C is missing:
        acquire the smallest useful information action
        return to contract compilation

    G <- seed a small graph from C and the observed state
    while budget remains:
        apply only decision-relevant operators
        A <- {a: Legal(a) and Viable(a)}
        if A is empty:
            return blocked with the unresolved contract and evidence
        P <- {a in A: Necessary(a) or Diagnostic(a)}
        a <- choose from P when P is nonempty;
             otherwise choose from A by cost, risk, and reversibility
        L <- compile a local contract with an expectation
        receipt <- execute a through the available host interface
        observation <- verify receipt against L and global Boundary

        if Boundary is violated:
            stop further side effects and report the violation
        if Accept(observation, receipt, L):
            close the action branch and update satisfied nodes
        if all Goal requirements and Evaluation evidence pass:
            return verified completion with receipts
        if observation contradicts the expectation:
            invalidate descendants with no independent satisfied support
            revise the affected branch and choose retry/replan/escalate/stop
```

```
        else:
            update G and continue

    return partial or blocked, with verified evidence and next decision
```

The semantic stopping condition is:

```
Goal(state) AND Boundary(trace) AND Evaluation(evidence)
```

Completion gate A branch closes when its declared observation, metric, and receipt satisfy `Accept`. Task completion additionally requires the contract's goal, boundary, and evidence predicates. Missing evidence yields an `unverified`, `partial`, or `blocked` state; a null, non-significant, or empty scientific result remains a valid completion when the declared analysis and receipt are complete.

Keep failed, partial, uncertain, and blocked states distinct from verified completion, and preserve negative results in the receipt.

Compact Handoff — A portable progress record

Prefer this record for progress or delegation handoffs:

```
Contract:       Goal / Boundary / Evaluation
Graph delta:    Added, removed, or invalidated nodes and edges
Action:         Selected action and why it is necessary or diagnostic
Receipt:        Observation and concrete evidence
Decision:       continue | prune | retry | replan | escalate | stop
Open risk:      Unresolved assumption or boundary/metric limitation
```

Common Mistakes — Failure mode and correction

Mistake	Correction
Acting while a missing fact could change the path	Take a bounded information action first.
Treating a written plan as authority	Re-run the omission and alternative checks.
Accepting `done` without a receipt	Mark it unverified and obtain the declared evidence.
Calling a metric pass success after a boundary violation	Stop and report the violation.
Continuing unsupported descendants after an unexpected observation	Invalidate descendants with no independent satisfied support and replan from the observed state.
Expanding the graph without changing a decision	Prune the graph and stop the information search.
Turning the method into a fixed checklist	Apply only the operators and ceremony with decision value.

The policy is instantiated through the host harness, model, tools, and task domain while retaining the same graph and operator semantics.

Minimal Example — Configuration change with evidence

For a request to change one configuration value and prove the repository stays usable:

```
Goal:       Change the named value and preserve a buildable repository.
Boundary:   Edit only the named file; no dependency or remote changes.
Evaluation: Targeted check passes; exact diff and file hash are recorded.

Graph:      requirement -> world_action -> evidence -> decision
            uncertainty -> information_action (locate the named file)

Expectation: one intended key changes and no unrelated file changes.
```

If the check fails or the diff is broader than expected, preserve the receipt, invalidate the completion edge, and replan from the actual state.

I DISCUSSION

An important direction for future work is to extend Meta-Graph Engineering from fixed task contracts to open-ended scientific investigations, where objectives, evidence standards, and hypotheses can evolve during execution. Such systems would need to learn and share execution structures across domains while preserving traceable links between discoveries, procedures, and evidence, a broad challenge beyond the present study. Progress will require long-horizon benchmarks and community standards that assess discovery quality, process validity, and reproducibility together.

J AI ASSISTANCE

In this work, generative AI tools were used to aid and polish the writing, including language editing, organization, and consistency checks. They were not used to generate data, formulate scientific claims, design experiments, run evaluations, or interpret the scientific results. The authors reviewed and finalized all AI-assisted text and remain responsible for the complete manuscript, including its claims, analyses, and artifacts.