EVALRESEARCHBENCH: CAN AI AGENTS DESIGN THEIR OWN EVALUATIONS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) relies on evaluation feedback to assess progress and guide further research, yet repeatedly running complex benchmarks is costly and slows iteration. Human experts reduce this cost by selecting benchmark subsets or designing compact suites. We ask whether AI agents can automate this design process and introduce EvalResearchBench (ERB), a benchmark for autonomous evaluation research. Given target materials, development references, candidate APIs, and fixed time and API budgets, an agent called the researcher selects or synthesizes tasks, implements graders, and revises them in pilot tests before freezing an executable evaluator for coding, co-work, and reasoning. We study 9 researchers and 13 candidate models and compare each frozen evaluator with 14 target benchmarks on score concordance and pairwise agreement. The best evaluators order about 75% of candidate pairs as the targets do, below the 91% ceiling set by disagreements among the targets. No researcher leads on every metric, and the best evaluator on development targets is not the best on sealed targets hidden from the researcher. A human-designed sample of public tasks remains a strong baseline, and the evaluator with the lowest recorded execution cost attains the highest pairwise agreement. Agents repair tasks and graders through pilot feedback, yet their evaluators can still truncate answers, exhaust the evaluation budget, or let a few questions dominate a domain score.

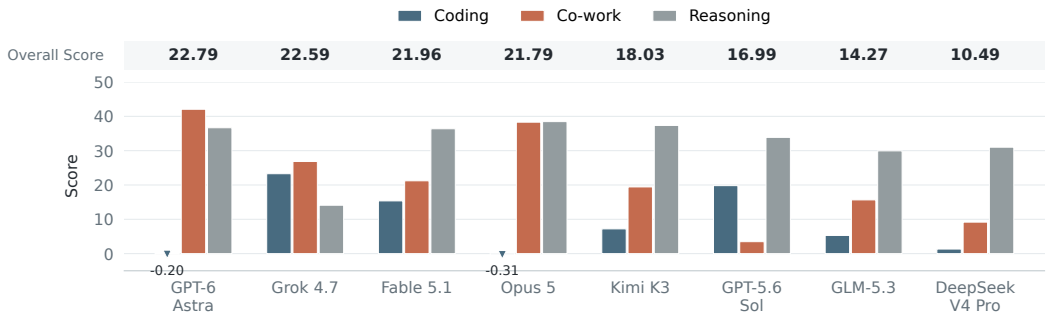

Figure 1: **Evaluation designs have distinct domain strengths.** Eight assessed programs, ordered by overall Score. Grouped bars share the same scale; overall averages all fourteen targets equally.

1 INTRODUCTION

Recursive self-improvement (RSI) relies on evaluation feedback to determine whether a change improves a system and which direction to explore next. Each new training recipe, dataset, or program creates another version to assess. Repeatedly running complex benchmarks incurs API and compute costs and delays the next experiment, particularly when evaluation requires long tool-using trajectories or specialized environments. Human experts reduce this burden by selecting representative subsets or designing cheaper proxies, as in tinyBenchmarks (Maia Polo et al., 2024). An autonomous improvement loop would need to perform this design step itself and obtain feedback that is informative enough to guide decisions and cheap enough to repeat.

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.

Recent work automates improvement while relying on supplied evaluation protocols. In PostTrainBench, agents choose data and post-training algorithms, but the target benchmark and evaluation harness are fixed (Rank et al., 2026). RSIBench-Data lets agents revise training-data strategies using checkpoint scores and trajectories from a fixed evaluation service (Meng et al., 2026). AlphaEvolve evolves programs using a user-provided evaluation function, with optional user-specified test cascades to screen proposals efficiently (Novikov et al., 2025). In each case, humans decide what to test, how to score it, and how to keep repeated measurement affordable, so extending autonomy to a new domain still requires a human-designed evaluation.

Other work automates parts of evaluation design. PACE selects inexpensive source questions and calibrates them against expensive agentic targets (Song et al., 2026). AutoBencher and BenchAgents automate benchmark construction and refinement (Li et al., 2025; Butt et al., 2025), and BenchBench compares the quality of model-designed suites (Zheng et al., 2026). None of these settings poses the full research problem, in which an agent chooses a measurement strategy, spends a limited experimental budget, and revises an executable evaluator until it tracks complex target benchmarks. Solving it requires inspecting target evidence, choosing or creating tasks, verifying graders, and deciding whether a pilot failure calls for harder questions, different coverage, a repaired rubric, or more execution budget. We therefore ask whether an agent with fixed time and API budgets can build an evaluator that reproduces how complex target benchmarks score and rank models. We call this capability autonomous evaluation research.

To address these gaps, we introduce **EvalResearchBench (ERB)**, in which the researcher designs the measurement itself, pays for every pilot from a fixed budget, and submits an executable evaluator that is scored against complex target benchmarks. A *researcher* is a model in an agent harness that designs the evaluator, a *candidate* is a model being tested, and a *target benchmark* is an established evaluation whose candidate scores the evaluator should reproduce. The researcher receives target descriptions, permitted tasks and grading resources, execution traces, development reference scores, and APIs for development candidates (Figure 2). In pilot tests, it observes candidate responses, costs, and agreement with the visible references, then revises its tasks, graders, weights, and answer allowances. It then freezes an evaluator that returns one score per domain. Sealed candidates and targets stay hidden during research, so the researcher can neither call those candidates nor see those reference scores. Independent assessment runs the frozen evaluator on all candidates under a separate budget and compares its scores with both development and sealed targets.

We evaluate 9 researcher configurations, each a model in the Codex or Claude Code harness run for one episode, on 13 open-weight and proprietary candidates and 14 target benchmarks in coding, co-work, and reasoning (Table 1). These domains cover the implementation, professional work, and analysis that autonomous research relies on. As a baseline, Random120 samples public tasks under fixed source quotas designed by humans.

We measure agreement in two ways. *Score* is 100 times Lin's concordance correlation coefficient between measured and reference scores (Lin, 1989), so it rewards matching score level and spread as well as order. *Pairwise agreement* (PA) is the fraction of candidate pairs that the evaluator orders as the target does. We also separate the one-time cost of building an evaluator from the cost of running it, which recurs for every new candidate.

No evaluator leads in every domain (Figure 1). Grok 4.7 leads coding, Astra leads co-work, and Opus 5 leads reasoning, while coding is the weakest domain for most evaluators. Astra attains the highest overall Score and Grok 4.7 the highest PA. Even the best PA, about 75%, stays well below the 91% ceiling set by disagreements among the targets. Fable 5.1 has the highest Score on development targets but not on sealed ones, so fitting visible references does not ensure transfer. Random120 remains a strong baseline. On identical candidates, seven of eight agent-designed evaluators exceed it in Score, but only four exceed it in PA. Evaluators with similar construction cost differ markedly in execution cost, and the one with the lowest recorded execution cost attains the highest PA. Agents repair tasks and graders through pilot feedback, yet their final evaluators can still truncate answers, exhaust the assessment budget, or let a few heavily weighted questions dominate a domain score.

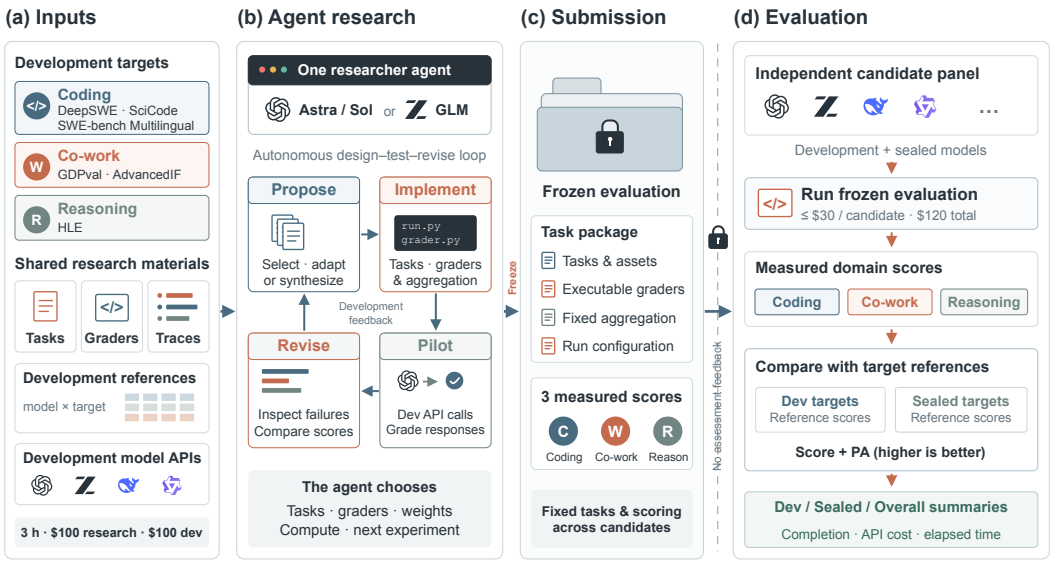

Figure 2: **EvalResearchBench.** Research materials support an autonomous design–test–revise loop. Tasks, graders, and score weights are frozen before independent assessment. Candidate-model and target visibility are assigned separately.

2 Related Work

2.1 Evaluation Design and Efficient Measurement

Evaluation design spans coverage and standardization (Ribeiro et al., 2020; Liang et al., 2023; Kiela et al., 2021), task construction (Perez et al., 2023; Liu et al., 2023a; Xia et al., 2024), and automated grading (Liu et al., 2023c; Zheng et al., 2023). AutoBencher refines datasets using model feedback (Li et al., 2025); BenchAgents integrates planning, generation, verification, and evaluation (Butt et al., 2025); BenchBench assesses benchmark designers through validity, discrimination, and ranking reliability (Zheng et al., 2026). Efficient measurement offers another route: APTBench transforms agent tasks and trajectories into lightweight questions (Qin et al., 2025), while tinyBenchmarks, Fluid Benchmarking, and Efficient Benchmarking of AI Agents study selection, estimation, and adaptive testing (Maia Polo et al., 2024; Hofmann et al., 2025; Ndzomga, 2026). Agent Psychometrics models task-level performance from task and system features (Ge et al., 2026). ERB evaluates an agent's ability to choose and develop measurement strategies through autonomous, budgeted experimentation.

2.2 Autonomous Research and Self-Improvement

Research benchmarks study iterative experimentation (Huang et al., 2024; Nathani et al., 2025), resource-bounded machine-learning research (Chan et al., 2025; Wijk et al., 2025), replication (Starace et al., 2025), and post-training (Rank et al., 2026). The AI Scientist and Agent Laboratory integrate multiple stages of scientific work (Lu et al., 2024; Schmidgall et al., 2025). Self-improvement systems such as FunSearch, AlphaEvolve, and the Darwin Gödel Machine use evaluation feedback to guide program or agent revisions (Romera-Paredes et al., 2024; Novikov et al., 2025; Zhang et al., 2026); MIRROR and TRIAGE study self-knowledge and resource allocation (Wang, 2026; Nazi & Dipta, 2026). ERB focuses on constructing the feedback instrument itself. Following artifact-based research benchmarks, it assesses the submitted evaluation independently of the process that produced it.

3 EVALRESEARCHBENCH

3.1 RESEARCH TASK

Researcher, candidates, and targets. The *researcher* is an LLM agent equipped with a harness that lets it inspect files, write and execute code, call models, and revise its work. A *candidate* is a model–execution configuration whose capabilities the researcher must measure. A *target benchmark* provides the reference candidate scores that the resulting evaluation should reproduce.

We cover three complementary capabilities needed for autonomous research. *Coding* supports implementing and testing systems, from repository engineering to scientific programming and terminal workflows (Huang et al., 2026; Tian et al., 2024a; Merrill et al., 2026). *Co-work* covers gathering information, following constraints, and producing professional deliverables (He et al., 2025; Wei et al., 2025; Patwardhan et al., 2025). *Reasoning* covers analytical work with academic questions, long documents, and specialized physics (Phan et al., 2025; Artificial Analysis, 2025a; Zhu et al., 2025). A single research episode designs an evaluation across all three domains, allowing the agent to share resources and choose where further testing is most useful.

Input materials. We provide target descriptions, available original tasks and grading resources, historical execution traces, and reference scores for development candidates. The researcher also receives metered candidate-model APIs, an execution SDK, and the submission contract. Resource availability varies by target; the agent may select existing tasks, synthesize new ones, or combine both approaches. Development tests return candidate responses and grading results for diagnosing errors and improving correlation with the visible target references.

Budgets. We constrain three costs: researcher inference pays for the agent's analysis and tool use; development testing pays for candidate and helper-model calls; independent assessment pays to execute the frozen design. Keeping the first two budgets separate prevents agent deliberation from consuming the testing allowance. A fixed deadline limits research time. Section 4.1 gives the numerical limits.

The researcher may select, synthesize, or simplify tasks, repair graders, change task weights, set answer allowances, and inspect pilot responses. Calls and retries pass through a metered gateway. The agent allocates its development budget across exploration, testing, and revision without a prescribed sequence.

Submission. To require a reusable test suite rather than a few demonstrations, the agent submits at least 100 distinct scored questions, together with executable graders and fixed aggregation rules. The program returns one measured score per domain from candidate responses and frozen assets, without candidate-identity lookup or access to reference labels during scoring. Questions, grading code, and weights are frozen before assessment. One ERB entry is one research episode by a particular model and harness.

3.2 INDEPENDENT ASSESSMENT

To test transfer beyond development feedback, we execute the frozen program independently on development and withheld candidates and compare its scores with visible and withheld target references. Candidate access and target-reference access are assigned separately (Table 1).

The executor records per-item rewards, termination reasons, and usage. A total assessment cap limits the cost of evaluating the candidate panel, while a per-candidate cap bounds the cost of applying the design to any one model. Assessment is charged separately from development, and its results are not returned to the researcher for further revision.

A graded response is complete under the submitted contract, including a wrong or empty answer at the output cap. Infrastructure failures and budget-stopped items remain incomplete. Let $C_{a,m,d}$ be the completed items for researcher a, candidate m, and domain d, where $s_{a,m,i}$ is the item grade and $w_{a,i} \geq 0$ is the weight assigned by the researcher to question i. Weights specify each question's

contribution to the domain score and are fixed before assessment. Its measured score is

$$x_{a,m,d} = \frac{\sum_{i \in C_{a,m,d}} w_{a,i} s_{a,m,i}}{\sum_{i \in C_{a,m,d}} w_{a,i}}. \tag{1}$$

A domain with no completed positive-weight item is missing. Coverage accompanies partial executions. Recorded expenditure separates researcher inference, development testing, and assessment, with unresolved reservations accounted for separately.

3.3 Score Agreement and Ordering

We assess both reference score levels and candidate ordering. We define *Score* as 100 times the concordance correlation coefficient (CCC) (Lin, 1989); pairwise agreement (PA) measures ordering.

For target t, match candidates with both an observed score $x_{a,m,d(t)}$ and a reference $y_{m,t}$. An eligible target has at least six matches and nonconstant references. Measured scores keep their native $[0, 1]$ grading scale. Percentage references are divided by 100; Elo references use the minimum and range of the fixed reference panel. These constants are shared across researchers (Appendix A). Each submitted domain score is compared unchanged with every eligible target in that domain.

Let $\bar{x}$, $\bar{y}$, $\mathrm{Var}(x)$, $\mathrm{Var}(y)$, and $\mathrm{Cov}(x, y)$ be population moments over matched candidates. Then

$$\mathrm{Score}_{a,t} = 100 \frac{2\,\mathrm{Cov}(x, y)}{\mathrm{Var}(x) + \mathrm{Var}(y) + (\bar{x} - \bar{y})^2}, \qquad \mathrm{Score}_{a,\mathcal{G}} = \frac{1}{|\mathcal{G}|} \sum_{t \in \mathcal{G}} \mathrm{Score}_{a,t}. \tag{2}$$

Exact agreement gives Score 100; constant measurements give zero, and opposing candidate differences can give negative values. CCC combines correlation with a penalty for mean and scale mismatch. Appendix A.1 analyzes compatibility among the target score scales.

For the non-tied reference pairs $\mathcal{P}_t = \{(i, j) : i < j,\ y_{i,t} \neq y_{j,t}\}$ among matched candidates, define

$$\mathrm{PA}_t = \frac{1}{|\mathcal{P}_t|} \sum_{(i,j) \in \mathcal{P}_t} \left[\mathbf{1}\{(x_{i,d} - x_{j,d})(y_{i,t} - y_{j,t}) > 0\} + \tfrac{1}{2} \mathbf{1}\{x_{i,d} = x_{j,d}\} \right]. \tag{3}$$

Correct order receives one point, a measured tie half a point, and a reversal zero. Reference ties are excluded; no significance threshold is applied. Both metrics average targets equally within each domain or visibility group; overall averages all eligible targets.

4 Experiments

4.1 Setup

We compare how different LLM researchers use the available materials and budget to build evaluations. The experiments first assess the resulting measurements, then examine the design choices and costs behind them. Appendix E traces how these choices arise during research.

Models. We study nine researcher configurations: GPT-6 Astra (OpenAI, 2026c), GPT-5.6 Sol (OpenAI, 2026b), GLM-5.3 (Z.ai, 2026b), DeepSeek V4 Pro (DeepSeek-AI et al., 2026), Kimi K3 (Kimi Team et al., 2026), Fable 5.1 (Anthropic, 2026b), Opus 5 (Anthropic, 2026c), Grok 4.7 (xAI, 2026), and Fable 5 (Anthropic, 2026a). Each contributes one research episode. The GPT researchers and Grok use Codex (OpenAI, 2025a); the other researchers use Claude Code (Anthropic, 2025). The candidate panel contains four development-visible and nine sealed models (Table 1).

Budgets and implementation. The research budget is three hours, $100 for researcher inference, and $100 for shared development/helper testing. Independent assessment has a $120 total cap and a $30 per-candidate cap. Pilots may use smaller suites than the final submission; the researcher allocates its development calls freely.

Table 1: **Experimental setup.** Three domains, thirteen candidates, and fourteen targets. Candidate and target visibility are independent; reference scores are fixed public snapshots.

Component	Development-visible	Sealed
Candidates	GPT-5.5; GLM-5.2; DeepSeek V4 Flash-0731; Qwen3.8 Max (4)	GPT-6 Astra; GPT-5.6 Sol; Kimi K3; MiMo V2.5 Pro; DeepSeek V3.2; Gemma 4 31B; Qwen3.5 397B-A17B; GPT-OSS 120B; Nemotron 3 Ultra (9)
Coding	DeepSWE; SciCode; SWE-bench Multilingual (3)	Terminal-Bench 2.1; ProgramBench (Vals); SWE-bench (Vals) (3)
Co-work	GDPval; AdvancedIF (2)	BrowseComp; Harvey LAB-AA; AA-Briefcase (3)
Reasoning	HLE (1)	AA-LCR; CritPt (2)
Research access	Candidate APIs, target descriptions, available development references, permitted tasks and grading resources	Candidate APIs and target references withheld from the researcher until after submission freeze

Public-task sampling baseline. A research agent should be compared with a simpler way to build a broad evaluation: sample questions from existing public sources. We use *Random120*, a source-stratified baseline that fixes the number of questions drawn from each source, ensuring coverage across task types before sampling. Specifically, we select sixteen public task sources, eligibility filters, source quotas, local graders, and equal item weighting within each domain. A seeded draw within each source selects 40 coding, 40 co-work, and 40 reasoning tasks without using target reference scores. We assess its twelve complete candidates against the same fourteen targets (Appendix B.1).

4.2 Main Results: Agreement with the Reference Panel

Table 2: **Frozen evaluation scores on development, sealed, and all targets.** Each group reports target-equal Score (100 times CCC) and PA (%) for coding, co-work, reasoning, and their target-equal average. Rows are ordered by overall Score. **Bold**/underlined mark the highest/second-highest observed values among the eight assessed artifacts. Random120 uses twelve complete candidates; the artifacts use thirteen. Fable 5 is a separate sensitivity scenario assigning 19 missing domain scores 100%.

		Development (6 targets)				Sealed (8 targets)				Overall (14 targets)			
Evaluator	Metric	Coding	Co-work	Reasoning	Avg.	Coding	Co-work	Reasoning	Avg.	Coding	Co-work	Reasoning	Avg.
GPT-6 Astra	Score↑	-5.19	**40.41**	60.64	20.98	4.79	43.16	24.69	24.15	-0.20	**42.06**	36.67	**22.79**
	PA↑	55.71	**75.76**	**83.33**	66.99	60.32	79.47	**76.09**	71.44	58.01	77.98	**78.51**	69.54
Grok 4.7	Score↑	**24.35**	10.34	13.33	17.84	**22.22**	37.84	14.48	**26.14**	**23.28**	26.84	14.10	22.59
	PA↑	**77.49**	72.10	82.22	**76.48**	66.85	81.63	71.93	73.66	**72.17**	77.81	75.36	**74.87**
Fable 5.1	Score↑	9.81	23.52	63.94	**23.40**	20.92	19.68	22.60	20.87	15.36	21.22	36.38	21.96
	PA↑	61.03	69.87	81.11	67.32	63.94	73.13	67.05	68.16	62.48	71.82	71.73	67.80
Opus 5	Score↑	-6.36	24.34	**77.99**	17.93	5.75	**47.60**	18.70	24.68	-0.31	38.30	**38.46**	21.79
	PA↑	51.26	69.02	80.00	61.97	56.30	74.61	71.89	67.06	53.78	72.37	74.59	64.88
Kimi K3	Score↑	6.49	12.34	55.33	16.58	7.93	24.16	**28.35**	19.12	7.21	19.43	37.35	18.03
	PA↑	64.33	74.07	81.11	70.38	**70.88**	**87.07**	73.99	**77.73**	67.61	**81.87**	76.36	74.58
GPT-5.6 Sol	Score↑	21.90	-6.66	50.45	17.14	17.71	10.24	25.58	16.88	19.81	3.48	33.87	16.99
	PA↑	66.94	46.38	81.11	62.45	60.60	60.31	74.54	63.98	63.77	54.74	76.73	63.32
GLM-5.3	Score↑	1.86	11.68	50.00	13.15	8.73	18.30	19.91	15.12	5.29	15.65	29.94	14.27
	PA↑	56.67	67.51	72.22	62.87	53.24	77.57	68.18	66.10	54.95	73.55	69.53	64.72
DeepSeek V4 Pro	Score↑	3.20	5.55	44.00	10.78	-0.53	11.59	24.51	10.28	1.33	9.17	31.01	10.49
	PA↑	56.78	64.27	74.44	62.22	63.19	65.91	64.18	64.46	59.99	65.25	67.60	63.50
Fable 5 (imputed)	Score↑	-7.77	-12.58	-18.63	-11.18	-5.68	-21.07	-18.58	-14.68	-6.72	-17.67	-18.60	-13.18
	PA↑	44.42	44.78	37.78	43.43	53.47	38.13	26.81	41.06	48.95	40.79	30.47	42.07
Random120	Score↑	21.44	-10.71	12.79	9.28	24.39	13.39	7.15	15.96	22.92	3.75	9.03	13.10
	PA↑	73.87	55.96	68.06	66.93	76.63	62.06	69.20	69.31	75.25	59.62	68.82	68.29

Eight submissions yield all 39 candidate-domain scores. Fable 5 reaches its assessment budget before completing the panel; Table 2 reports its stated imputation scenario separately from the eight observed-score results.

Overall results and domain strengths. Astra leads overall Score, whereas Grok leads PA: reproducing score levels and recovering candidate order favor different designs. Domain strengths also vary, with no researcher dominating both metrics across all three domains. Decomposing CCC as rC_b helps explain why: Astra and GLM-5.3 have similar co-work correlations, but Astra aligns more closely with the reference scale (Table 10). A useful ordering alone does not guarantee faithful score reproduction.

Development and sealed targets reveal different strengths. Fable 5.1 leads development Score, but Grok leads sealed Score. This reversal makes cross-target transfer a distinct requirement for evaluation research: a useful design must preserve capability signals beyond the references used to develop it. Systematic score shifts across targets within the same domain (Figure 7) further show why a strong fit to one benchmark does not establish a broadly useful evaluator.

Public-task sampling is a strong baseline. Random120's public sources, domain quotas, and graders encode task-design knowledge, making it a strong baseline, especially in coding. Its coding and reasoning coverage recovers much of the candidate ordering, while Astra's Score gains concentrate in co-work and reasoning (Figure 3). DeepSeek V4 Pro illustrates why iteration alone need not beat this baseline. It revises coding tasks and increases reasoning allowances, yet its short, deterministic coding and co-work checks yield scores clustered near the ceiling; its reasoning gains cannot offset the coding deficit. Research adds value when it identifies tasks that preserve target-relevant differences between candidates. Ease of grading and repeated revision do not by themselves ensure informative measurements.

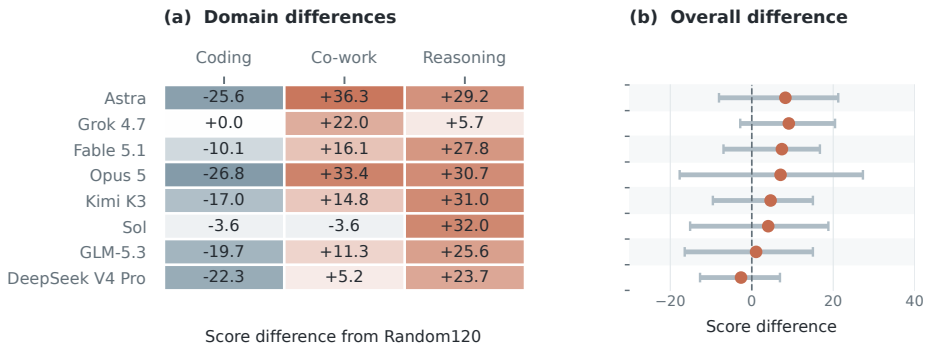

Figure 3: **Where research differs from fixed public-task sampling.** Differences from Random120 on identical candidate–target intersections: (a) per domain; (b) overall, with central 95% candidate-resampling ranges. The twelve candidates are jointly resampled; fourteen targets and frozen evaluations stay fixed.

4.3 What Shapes Evaluation Quality?

We analyze three design decisions—task selection, question count, and aggregation—by varying one part of each evaluation while holding candidate responses and references fixed.

Choose tasks for their measurement contribution. K3 adds hard coding problems after near-ceiling pilots; Astra adds repository traces after inspecting upstream software tasks. Removing these groups, however, has only modest effects relative to same-domain, same-size random deletions. K3's addition improves Score but slightly reduces PA; Astra's repository group leaves PA unchanged. By contrast, Astra's spreadsheet tasks contribute strongly to both co-work metrics (Figure 4a,b). Task realism and difficulty motivate a design, but its value depends on how it separates candidates in relation to the target.

Many Astra spreadsheet responses exhaust their answer allowance, so the observed separation reflects the task under its chosen execution conditions. Grok responds to truncated hard-task pilots by favoring shorter, self-contained checks and adjusting answer allowances. DeepSeek instead expands

reasoning allowances and selects short exact-answer items. A short expected answer does not itself make a problem easy to solve.

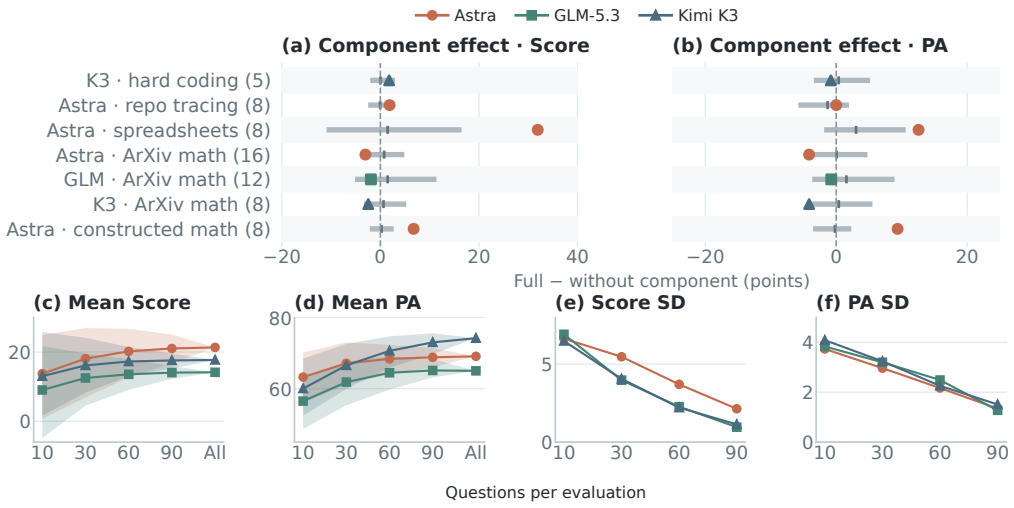

Figure 4: **Task contribution and measurement variability.** (a,b) Full-minus-deleted Score and PA for seven task groups; parentheses give question counts. Gray intervals cover the central 95% of 1,000 same-domain, same-size random deletions. (c,d) Mean overall Score and PA across nested question subsets, with central 95% bands. (e,f) Their standard deviations. Candidates, targets, and responses are fixed.

More questions improve both average quality and consistency. We draw nested, domain-stratified subsets from the frozen Astra, GLM-5.3, and K3 suites and recompute their scores on the same candidate panel. As subsets grow, mean Score rises and both Score and PA fluctuate less across question draws (Figure 4c–f). Small subsets can omit useful task families or overrepresent a few responses, even when their domain proportions are fixed. The design problem is to retain this signal and consistency with fewer calls.

Use weights to preserve more than a development ordering. Grok groups pilot questions by which development candidates pass, then searches for weights that reproduce the visible ordering and score gaps. For co-work, it assigns most weight to questions that all three tested development candidates pass, using a small remainder to shape their differences. Consequently, six of its twenty co-work questions carry almost all scoring weight. Fable 5.1 distributes influence more evenly, with a modest upweighting of professional tasks (Figure 5a).

Recomputing Grok's scores with equal item weights on the same responses substantially improves coding and co-work Score, while overall PA barely changes (Figure 5). The Score improvement also holds on the complete-item candidate subset. The frozen suite therefore contains useful measurement signal that its chosen weights underuse. Evaluation design must consider task contribution, sampling variability, and score aggregation jointly.

4.4 What Makes an Evaluation Cost-Efficient?

Low execution cost can preserve measurement quality. An evaluator is built once but may be run repeatedly, so we separate construction cost from frozen assessment cost. Grok combines low assessment cost with the highest overall PA and a competitive Score (Figure 6; Table 2). Grok and Fable 5.1 submit similar question counts, yet their execution costs differ substantially. Task count alone therefore gives little indication of an evaluator's recurring cost.

Pilot feedback shapes answering and grading costs. Grok's hard HLE pilots exhaust their answer allowance without producing visible answers. It responds by favoring short code, output predictions, and numeric checks with local graders. Fable 5.1 retains longer professional deliverables

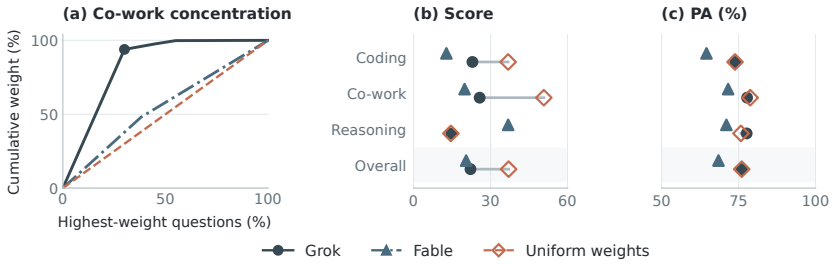

Figure 5: **Weight allocation and measurement quality.** Grok and Fable (Fable 5.1) use their submitted weights. (a) Cumulative co-work weight versus the fraction of questions, sorted by decreasing weight; the dashed diagonal denotes uniform weighting. (b,c) Score and PA on the same twelve-candidate, fourteen-target panel. Here *Uniform weights* reweights Grok's grades only; connectors show this change, with its responses and item exclusions fixed.

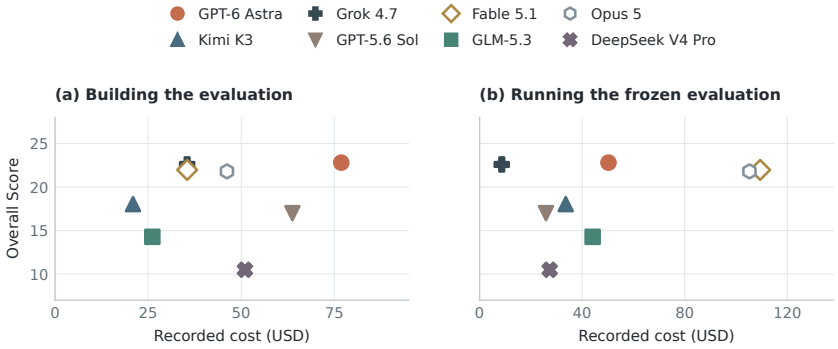

Figure 6: **Spending and measurement quality.** Overall Score against recorded API-equivalent costs for (a) researcher inference plus development tests and (b) frozen assessment. Solid markers use metered usage; hollow markers use collaborator reports. Unresolved reservations are excluded.

and increases their answer allowance after truncated pilots; its rubric judge adds a substantial second expense. Opus incurs little grading expense but still spends heavily on candidate answers. These trajectories identify two distinct design choices behind execution cost: what a candidate must produce and how its answer is judged (Appendix A.3).

5 Case Study: How Researchers Revise Evaluations

Researchers use development feedback to expand coverage, repair tasks, and adjust execution. Astra develops repository-tracing tasks from source inspection, resolves prompt–rubric mismatches through pilot feedback, and adjusts execution after truncated responses. GLM corrects invalid examples and numerical answers, while K3 adds harder coding tasks after near-ceiling pilot scores. Section 4.3 analyzes these design choices; Appendix E details the cases.

6 Conclusion

ERB measures autonomous evaluation research under fixed budgets. Independent assessment separates score reproduction, ordering, and execution cost. Source inspection and pilot audits reveal how agents revise tasks and graders; component and subset analyses show why difficulty, nominal size, and measurement value must be assessed separately. This establishes an executable benchmark for the evaluation-design capability that autonomous improvement requires.

AI USE STATEMENT

Generative AI tools assisted with research-methodology feedback, software implementation, analysis of execution traces, interpretation of results, and figure preparation. These uses are separate from the researcher agents and candidate models evaluated as experimental subjects. The reported numerical analyses use saved measurements and executable analysis scripts. The authors are responsible for the final text, claims, code, and artifacts, including AI-assisted contributions.

ETHICS STATEMENT

This work studies automated evaluation design rather than experiments involving human participants. Generated tasks and graders can contain errors or encode biases; agreement with public reference scores does not establish broad validity or safety. Such evaluations should not be the sole basis for consequential deployment decisions. Credentials and private access information are excluded from the manuscript; redistribution of third-party tasks and traces remains subject to their source terms.

REPRODUCIBILITY STATEMENT

Section 3 specifies the research task, the submission contract, and independent assessment, and Section 3.3 defines the measured domain scores, Score, and PA. Section 4.1 and Table 1 list the researcher configurations and their harnesses, the candidate panel, the development and sealed splits, and all budgets. Appendix A identifies the public leaderboard behind every reference score, the fixed reference scales, the candidate models, and per-target results; Appendix A.2 describes the cost accounting. Appendix B specifies the construction of Random120, including its sources, quotas, and sampling seed, together with the question-subset and candidate-resampling protocols, and Appendix C specifies the component-deletion and reweighting analyses. Appendix D describes the research instructions and the task interface given to researchers. All analyses beyond the frozen assessments reuse saved responses and make no new model calls.

REFERENCES

Anthropic. Claude Code. https://code.claude.com/docs/en/overview, 2025.

Anthropic. System card: Claude Fable 5 & Claude Mythos 5. https://www.anthropic.com/claude-fable-5-mythos-5-system-card, 2026a.

Anthropic. System card: Claude Fable 5.1 & Claude Mythos 5.1. https://www.anthropic.com/claude-fable-5-1-mythos-5-1-system-card, 2026b.

Anthropic. System card: Claude Opus 5. https://www.anthropic.com/claude-opus-5-system-card, 2026c.

Artificial Analysis. Announcing Artificial Analysis Long Context Reasoning (AA-LCR). https://artificialanalysis.ai/articles/announcing-aa-lcr, 2025a.

Artificial Analysis. CritPt benchmark leaderboard. https://artificialanalysis.ai/evaluations/critpt, 2025b.

Artificial Analysis. Announcing AA-Briefcase: A frontier knowledge work evaluation. https://artificialanalysis.ai/articles/aa-briefcase, 2026a.

Artificial Analysis. AA-Briefcase v1.1: Agentic knowledge work benchmark. https://artificialanalysis.ai/evaluations/aa-briefcase, 2026b.

Artificial Analysis. Harvey LAB-AA benchmark leaderboard. https://artificialanalysis.ai/evaluations/harvey-lab-aa, 2026c.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating LLMs on uncontaminated math competitions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=y0zL9IZxZ7.

Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. BenchAgents: Multi-Agent Systems for Structured Benchmark Creation. *arXiv preprint arXiv:2410.22584*, 2025. URL https://arxiv.org/abs/2410.22584.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e3767db483c942b883eb4f8cfb74e31-Abstract-Conference.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Jasper Dekoninck, Tim Gehrunger, and Martin Vechev. ArXivMath: Evaluating LLMs on mathematical research problems from recent arXiv papers. https://matharena.ai/arxivmath/, 2026.

Chris Ge, Daria Kryvosheieva, Daniel Fried, Uzay Girit, and Kaivalya Hariharan. Agent psychometrics: Task-level performance prediction in agentic coding benchmarks. *arXiv preprint arXiv:2604.00594*, 2026. URL https://arxiv.org/abs/2604.00594.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

Harvey. Introducing Harvey's Legal Agent Benchmark. https://www.harvey.ai/blog/introducing-harveys-legal-agent-benchmark, 2026.

Yun He, Wenzhe Li, Hejia Zhang, et al. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. *arXiv preprint arXiv:2511.10507*, 2025. URL https://arxiv.org/abs/2511.10507.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Valentin Hofmann, David Heineman, Ian Magnusson, Kyle Lo, Jesse Dodge, Maarten Sap, Pang Wei Koh, Chun Wang, Hannaneh Hajishirzi, and Noah A. Smith. Fluid Language Model Benchmarking. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=mxcCg9YRqj.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. *arXiv preprint arXiv:2310.03302*, 2024. URL https://arxiv.org/abs/2310.03302.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026. URL https://arxiv.org/abs/2607.07946.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

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

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking Benchmarking in NLP. *arXiv preprint arXiv:2104.14337*, 2021. URL https://arxiv.org/abs/2104.14337.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench hard: Unrestrained reasoning about action, change, and planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=WIXohR7mEo.

Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. AutoBencher: Towards Declarative Benchmark Construction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/eb216114f3eaad22506fd1bc7bbff0ca-Abstract-Conference.html.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic Evaluation of Language Models. *arXiv preprint arXiv:2211.09110*, 2023. URL https://arxiv.org/abs/2211.09110.

Lawrence I-Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45(1):255–268, 1989. doi: 10.2307/2532051.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning, 2020. URL https://arxiv.org/abs/2007.08124.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. *arXiv preprint arXiv:2305.01210*, 2023a. URL https://arxiv.org/abs/2305.01210.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems, 2023b. URL https://arxiv.org/abs/2306.03091.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023c. URL https://aclanthology.org/2023.emnlp-main.153/.

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.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinyBenchmarks: evaluating LLMs with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34303–34326. PMLR, 2024. URL https://proceedings.mlr.press/v235/maia-polo24a.html.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. RSIBench-Data: Benchmarking data-centric research for recursive self-improvement. *arXiv preprint arXiv:2607.25886*, 2026. URL https://arxiv.org/abs/2607.25886.

Mercor. APEX benchmarks: AI productivity index leaderboards. https://www.mercor.com/apex/, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/abs/2601.11868.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A New Framework and Benchmark for Advancing AI Research Agents. *arXiv preprint arXiv:2502.14499*, 2025. URL https://arxiv.org/abs/2502.14499.

Zabir Al Nazi and Shubhashis Roy Dipta. TRIAGE: Evaluating prospective metacognitive control in LLMs under resource constraints. *arXiv preprint arXiv:2605.13414*, 2026. URL https://arxiv.org/abs/2605.13414.

Franck Ndzomga. Efficient benchmarking of AI agents. *arXiv preprint arXiv:2603.23749*, 2026. URL https://arxiv.org/abs/2603.23749.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

NVIDIA, Aaron Blakeman, Aaron Thomas, Aastha Jhunjhunwala, Abhibha Gupta, Abhinav Khattar, Adam Rajfer, Adi Renduchintala, Adil Asif, Aditya Vavre, et al. Nemotron 3 Ultra: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. *arXiv preprint arXiv:2606.15007*, 2026. URL https://arxiv.org/abs/2606.15007.

OpenAI. Codex CLI. https://github.com/openai/codex, 2025a.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025b. URL https://arxiv.org/abs/2508.10925.

OpenAI. GPT-5.5 system card. https://deploymentsafety.openai.com/gpt-5-5, 2026a.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026b.

OpenAI. GPT-6 Astra system card. https://deploymentsafety.openai.com/gpt-6-astra, 2026c.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, et al. GDPval: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025. URL https://arxiv.org/abs/2510.04374.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Ben Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering Language Model Behaviors with Model-Written Evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13387–13434, 2023. doi: 10.18653/v1/2023.findings-acl.847. URL https://aclanthology.org/2023.findings-acl.847/.

Long Phan, Alice Gatti, Ziwen Han, et al. Humanity's Last Exam. *arXiv preprint arXiv:2501.14249*, 2025. URL https://arxiv.org/abs/2501.14249.

Jiarui Qin, Yunjia Xi, Junjie Huang, Renting Rui, Di Yin, Weiwen Liu, Yong Yu, Weinan Zhang, and Xing Sun. APTBench: Benchmarking agentic potential of base LLMs during pre-training. *arXiv preprint arXiv:2510.24397*, 2025. URL https://arxiv.org/abs/2510.24397.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13025–13048, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026a.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, 2026b.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM Agents Automate LLM Post-Training? *arXiv preprint arXiv:2603.08640*, 2026. URL https://arxiv.org/abs/2603.08640.

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, 2023. URL https://arxiv.org/abs/2311.12022.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://aclanthology.org/2020.acl-main.442/.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM Agents as Research Assistants. *arXiv preprint arXiv:2501.04227*, 2025. URL https://arxiv.org/abs/2501.04227.

Yueqi Song, Lintang Sutawika, Jiarui Liu, Lindia Tjuatja, Jiayi Geng, Yunze Xiao, Daniel Lee, Aditya Bharat Soni, Vincent Lo, Xiang Yue, and Graham Neubig. PACE: A proxy for agentic capability evaluation. *arXiv preprint arXiv:2607.02032*, 2026. URL https://arxiv.org/abs/2607.02032.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's Ability to Replicate AI Research. *arXiv preprint arXiv:2504.01848*, 2025. URL https://arxiv.org/abs/2504.01848.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, et al. SciCode: A research coding benchmark curated by scientists. *arXiv preprint arXiv:2407.13168*, 2024a. URL https://arxiv.org/abs/2407.13168.

Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Yinxu Pan, Yesai Wu, Hui Haotian, Liu Weichuan, Zhiyuan Liu, et al. Debugbench: Evaluating debugging capability of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4173–4198, 2024b.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=YXogl4uQUO.

Vals AI. ProgramBench. https://www.vals.ai/benchmarks/programbench, 2026a.

Vals AI. SWE-bench Verified. https://www.vals.ai/benchmarks/swebench, 2026b.

Jason Z Wang. MIRROR: A hierarchical benchmark for metacognitive calibration in large language models. *arXiv preprint arXiv:2604.19809*, 2026. URL https://arxiv.org/abs/2604.19809.

Jason Wei, Zhiqing Sun, Spencer Papay, et al. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating

frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. Lifbench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16445–16468, 2025.

xAI. Model card: Grok 4.7. https://media.x.ai/v1/website/card4p7-3a96f40b.pdf, 2026.

Chunqiu Steven Xia, Yinlin Deng, and Lingming Zhang. Top Leaderboard Ranking = Top Coding Proficiency, Always? EvoEval: Evolving Coding Benchmarks via LLM. *arXiv preprint arXiv:2403.19114*, 2024. URL https://arxiv.org/abs/2403.19114.

Xiaomi MiMo Team. Xiaomi MiMo-V2.5-Pro. https://mimo.xiaomi.com/mimo-v2-5-pro, 2026.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025. URL https://arxiv.org/abs/2504.21798.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, Diyi Yang, and Ofir Press. Programbench: Can language models rebuild programs from scratch?, 2026. URL https://arxiv.org/abs/2605.03546.

Z.ai. GLM-5.2: Built for long-horizon tasks. https://z.ai/blog/glm-5.2, 2026a.

Z.ai. GLM-5.3: Frontier coding with emergent cyber capabilities. https://z.ai/blog/glm-5.3, 2026b.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

Yandan Zheng, Haoran Luo, Zhenghong Lin, Wenjin Liu, and Anh Tuan Luu. BenchBench: Benchmarking Automated Benchmark Generation. *arXiv preprint arXiv:2603.20807*, 2026. URL https://arxiv.org/abs/2603.20807.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Minhui Zhu, Minyang Tian, Xiaocheng Yang, Tianci Zhou, Lifan Yuan, Penghao Zhu, Eli Chertkov, Shengyan Liu, Yufeng Du, Ziming Ji, et al. Probing the Critical Point (CritPt) of AI Reasoning: a Frontier Physics Research Benchmark. *arXiv preprint arXiv:2509.26574*, 2025. URL https://arxiv.org/abs/2509.26574.

A TARGET-LEVEL ASSESSMENT AND SCORE SCALES

Table 2 presents the complete development, sealed, and overall breakdown in the main text. This appendix gives target-level results and the reference-scale conventions.

Assessment tables use Score and PA from Section 3.3, with target-equal aggregation and the same reference-bearing candidate IDs per target. Domain columns average their own targets. Frozen-component and question-count analyses use these same two metrics on a common twelve-candidate panel.

Measured scores stay on their frozen $[0, 1]$ grading scale. Percentage references use offset zero and divisor 100, including GDPval, Harvey criterion pass rate, ProgramBench strict resolution, and CritPt. AA-Briefcase is an Elo reference and uses the following fixed panel constants, shared by all artifacts:

Reference	Offset	Divisor
AA-Briefcase Elo	0.00	1640.35

The Elo constants are the minimum and range over the frozen reference-bearing candidate panel, shared across all researchers. This convention fixes the unit of score differences for the panel.

The development group contains DeepSWE (Huang et al., 2026), SciCode (Tian et al., 2024a), SWE-bench Multilingual (Yang et al., 2025), AdvancedIF (He et al., 2025), GDPval (Patwardhan et al., 2025), and HLE (Phan et al., 2025). The sealed group contains Terminal-Bench 2.1 (Merrill et al., 2026), BrowseComp (Wei et al., 2025), AA-LCR (Artificial Analysis, 2025a), ProgramBench (Vals) (Yang et al., 2026), SWE-bench (Vals) (Jimenez et al., 2024), Harvey LAB-AA (Harvey, 2026), AA-Briefcase (Artificial Analysis, 2026a), and CritPt (Zhu et al., 2025). The criterion comprises published score snapshots from their source protocols. Nine reference sets come from Mercor's APEX open-source leaderboards (Mercor, 2026): DeepSWE, SciCode, SWE-bench Multilingual, Terminal-Bench 2.1, AdvancedIF, GDPval, BrowseComp, HLE, and AA-LCR. ProgramBench (Vals) and SWE-bench (Vals) come from Vals AI (Vals AI, 2026a;b), and Harvey LAB-AA, AA-Briefcase, and CritPt from Artificial Analysis (Artificial Analysis, 2026c;b; 2025b).

The four development-visible candidates are GPT-5.5 (OpenAI, 2026a), GLM-5.2 (Z.ai, 2026a), DeepSeek V4 Flash-0731 (DeepSeek-AI et al., 2026), and Qwen3.8 Max (Qwen Team, 2026b). The nine sealed candidates are GPT-6 Astra (OpenAI, 2026c), GPT-5.6 Sol (OpenAI, 2026b), Kimi K3 (Kimi Team et al., 2026), MiMo V2.5 Pro (Xiaomi MiMo Team, 2026), DeepSeek V3.2 (DeepSeek-AI et al., 2025), Gemma 4 31B (Gemma Team et al., 2026), Qwen3.5 397B-A17B (Qwen Team, 2026a), GPT-OSS 120B (OpenAI, 2025b), and Nemotron 3 Ultra (NVIDIA et al., 2026).

A.1 REPRESENTATIVE TARGET SCATTERPLOTS

Figure 7 shows one development and one sealed target in each domain for Astra, K3, and Fable 5.1. Astra obtains strong co-work agreement on both GDPval (Score 55.78) and BrowseComp (42.23). In reasoning, the three evaluators reach Scores of 55.33–63.94 on HLE, but only 3.62–7.83 on AA-LCR, where their measured scores lie systematically below the references. A single domain score can therefore preserve different aspects of different targets.

Incompatibility of two coding references. A single domain score is compared with several target references, which need not agree in level or ordering. We first quantify how this disagreement limits raw-score reproduction. ProgramBench (Vals) and SWE-bench Multilingual share six reference-bearing candidates. Let $y_{m,1}$ and $y_{m,2}$ be their fixed-scale reference scores and x_m a single coding score for candidate m. The midpoint $x_m = (y_{m,1} + y_{m,2})/2$ minimizes pooled squared error to both references, giving

$$\min_{x_1,\ldots,x_6} \sqrt{\frac{1}{12}\sum_{m=1}^{6}\sum_{j=1}^{2}(x_m - y_{m,j})^2} = \frac{1}{2}\sqrt{\frac{1}{6}\sum_{m=1}^{6}(y_{m,1} - y_{m,2})^2} = 0.4356. \tag{4}$$

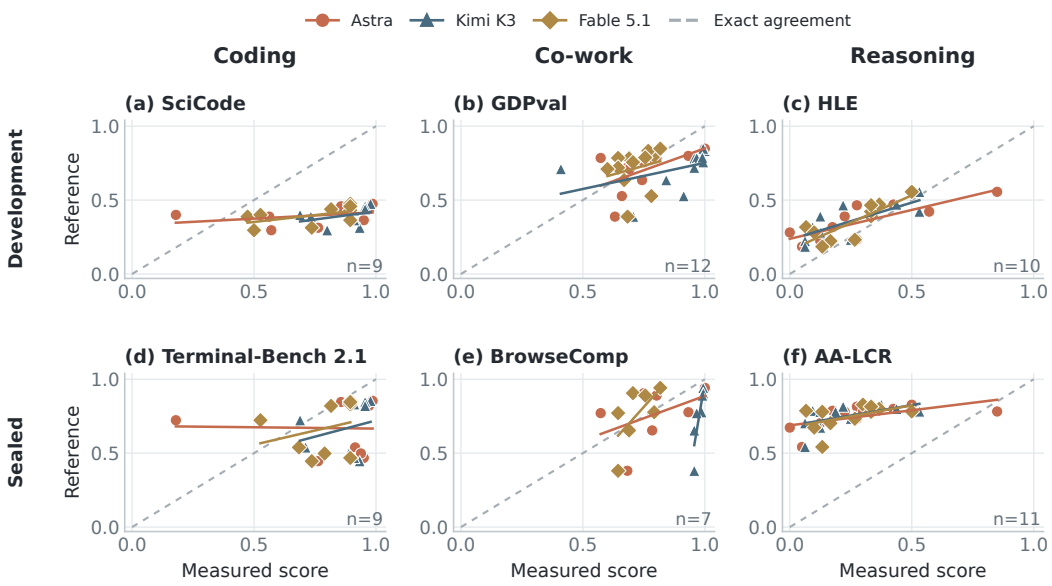

Figure 7: **Measured scores versus target references.** Columns show coding, co-work, and reasoning; rows show one development and one sealed target per domain. Points denote candidates, colored lines show within-panel OLS trends, and the dashed line marks exact agreement. Score and PA are computed from the original points.

The reference means are 0.0075 and 0.876 on the same six candidates. The resulting raw-score error floor belongs to the reference pair; CCC also depends on covariance and dispersion. PA separately reports ordering.

The same calculation extends to every reference in a domain. For target set T_d, minimize the target-equal mean squared error $|T_d|^{-1}\sum_{t\in T_d}|\mathcal{M}_t|^{-1}\sum_{m\in\mathcal{M}_t}(x_m - y_{m,t})^2$ over one common $x_m \in [0,1]$ per candidate. Its exact minimizer is the weighted mean of that candidate's available reference scores, with target t weighted by $1/|\mathcal{M}_t|$. The resulting oracle RMSE floors are:

Domain	Development	Sealed	Overall
Coding	0.209	0.353	0.303
Co-work	0.074	0.088	0.128
Reasoning	0.000	0.284	0.244

These label-informed values characterize the reference panel's raw-score compatibility. The zero for single-target development reasoning follows from assigning its reference score directly; the RMSE floors are distinct from CCC ceilings.

Compatible-reference sensitivity. To examine how reference-scale differences affect the results, we form an exploratory target subset without reading any researcher score: in each domain, choose the largest group in which every pair shares at least six reference-bearing candidates and has reference-score RMSE at most 0.25; break ties by the group's mean pairwise RMSE. This selects SWE-bench Multilingual, SWE-bench (Vals), and Terminal-Bench 2.1 for coding; AdvancedIF, BrowseComp, and GDPval for co-work; and HLE and CritPt for reasoning. Four selected targets are development-visible and four are sealed. Target-equal Score and PA on this eight-target subset are:

Frozen artifact	Score ↑	PA (%) ↑
Fable 5.1	33.47	69.57
Astra	27.51	67.08
Grok	23.95	75.06
Sol	22.31	59.82
Opus 5	22.29	62.73
K3	22.12	73.08
DeepSeek V4 Pro	17.46	65.00
GLM-5.3	17.32	61.28

This scale-based sensitivity checks how target composition affects raw-score concordance. Fable 5.1 leads the eight-target subset, while Astra leads the primary fourteen-target panel.

Reference-imposed ordering limit. Reference targets in one domain sometimes order the same candidate pair differently. We calculate the largest PA attainable by *any* single strict candidate ordering per domain on the fixed panel. Every target and every non-tied pair within it receives exactly the weight used in Section 3.3; an exact subset dynamic program maximizes the weighted agreement. A measured tie earns the average of its two possible orientations and cannot improve the optimum. This label-informed oracle is an upper bound for a frozen evaluator, not a deployable score:

Domain	Development	Sealed	Overall
Coding	92.11%	94.72%	92.36%
Co-work	84.68%	95.40%	88.72%
Reasoning	100.00%	93.18%	93.03%
All targets	90.95%	94.59%	91.20%

Each column optimizes its own target set, so the overall maximum need not average the development and sealed maxima. Cross-target order conflicts cap overall PA at 91.20%, compared with the best observed artifact's 74.87%.

Fable 5 produced 20 of 39 candidate-domain reports before assessment stopped at the budget cap. The main ranking excludes it. Table 2 separately reports the fixed scenario that assigns its 19 unobserved domain scores a value of one; these are scenario-conditional metrics, not observed-panel estimates or metric bounds.

Fixed-score sanity check. To check whether an uninformative evaluator can obtain a favorable metric, we assign every candidate a score of 0.5 in every domain without making model calls. This yields overall Score$= 0$ and PA$= 50.00\%$ on the same fourteen target panels. All measured pairs tie. Score requires score variation aligned with the reference and penalizes mean and scale mismatch.

A.2 Cost accounting

Costs separate researcher inference, shared development/helper testing, and frozen assessment. The values are standardized API-equivalent estimates.

A.3 Task design, execution cost, and score concentration

We compare frozen task forms, scoring weights, and recorded execution cost for the suites discussed in Section 4.4.

Table 4 separates candidate generation from rubric and equivalence grading costs.

Grok's short functions, program-output predictions, scalar answers, and format checks contrast with Fable 5.1's scientific code, professional tasks, and HLE questions. Grok's graders execute code or compare explicit answers and formats.

Target	Group	n	Metric	Astra	Grok	Fable 5.1	Opus 5	K3	Sol
DeepSWE	Dev	9	PA↑	53.03	84.85	59.09	50.00	63.64	78.79
			Score↑	-2.32	21.48	8.61	-1.08	2.76	14.54
SciCode	Dev	9	PA↑	69.44	83.33	72.22	68.06	72.22	66.67
			Score↑	5.73	2.75	7.54	7.39	1.80	2.15
SWE-bench Multilingual	Dev	8	PA↑	44.64	64.29	51.79	35.71	57.14	55.36
			Score↑	-18.99	48.81	13.27	-25.37	14.91	49.01
Terminal-Bench 2.1	Sealed	9	PA↑	55.56	75.00	66.67	50.00	63.89	58.33
			Score↑	-1.85	26.70	18.99	-21.84	10.64	20.44
ProgramBench (Vals)	Sealed	8	PA↑	68.75	50.00	56.25	66.67	85.42	64.58
			Score↑	0.53	-0.04	0.32	0.59	0.24	0.12
SWE-bench (Vals)	Sealed	10	PA↑	56.67	75.56	68.89	52.22	63.33	58.89
			Score↑	15.69	39.99	43.45	38.49	12.90	32.59
AdvancedIF	Dev	11	PA↑	66.67	63.89	68.52	59.26	64.81	35.19
			Score↑	25.03	-4.61	28.14	-0.67	-1.00	-18.32
GDPval	Dev	12	PA↑	84.85	80.30	71.21	78.79	83.33	57.58
			Score↑	55.78	25.28	18.90	49.36	25.68	5.01
BrowseComp	Sealed	7	PA↑	71.43	85.71	80.95	76.19	95.24	57.14
			Score↑	42.23	32.32	39.52	28.61	6.06	-8.04
Harvey LAB-AA	Sealed	9	PA↑	83.33	79.17	63.89	69.44	77.78	58.33
			Score↑	42.54	32.26	3.72	58.76	31.02	10.50
AA-Briefcase	Sealed	11	PA↑	83.64	80.00	74.55	78.18	88.18	65.45
			Score↑	44.73	48.95	15.81	55.43	35.40	28.24
HLE	Dev	10	PA↑	83.33	82.22	81.11	80.00	81.11	81.11
			Score↑	60.64	13.33	63.94	77.99	55.33	50.45
AA-LCR	Sealed	11	PA↑	78.70	70.37	66.67	74.07	72.22	74.07
			Score↑	7.83	19.18	3.62	5.67	4.29	3.80
CritPt	Sealed	12	PA↑	73.48	73.48	67.42	69.70	75.76	75.00
			Score↑	41.55	9.78	41.58	31.73	52.42	47.36

Table 3: Per-target PA (%) and Score for the four highest overall-Score artifacts, plus K3 and Sol. Scores use the frozen metrics and target panels in Section 3.3; n is the number of reference-bearing candidates.

Artifact	Questions	Candidate	Rubric judge	Equivalence	Total
Grok 4.7	107	8.63	–	–	8.63
Fable 5.1	110	75.44	33.97	0.01	109.41
Fable 5	118	100.00	18.02	<0.01	118.02
Opus 5	108	105.22	–	0.02	105.24

Table 4: **Where the execution budget goes.** Recorded assessment expenditure in USD, split by candidate and grader handles. Questions denote declared suite size; costs are incurred run totals, including partial executions, rather than equal-workload estimates. Unresolved reservations are excluded.

Grok's prompts have median length 233 characters and use no assessment helper calls. The separate Fable 5 artifact has longer professional tasks and output allowances, alongside higher recorded assessment cost.

Separate Fable 5 candidate-cost comparison. For GPT-5.5 and Astra, Fable 5's recorded candidate-call cost is 16.73 and 7.12 times Grok's, while its completed-item counts differ by less than 10% (Table 6).

Question count versus influence on the score. For normalized nonnegative domain weights w_i, define the weight-equivalent count

$$N_w = \frac{1}{\sum_i w_i^2}, \qquad \sum_i w_i = 1.$$

It equals the question count under uniform weights and decreases with concentration. Grok's coding/co-work/reasoning counts are 62/20/25, but N_w is 21.84/6.78/19.07. Fable 5.1's corresponding values are 38/40.32/30 for 38/42/30 questions.

Grok's six highest-weight co-work items carry 93.87% of that domain score (Figure 12). Low-weight questions still incur execution cost.

Dimension	Grok	Fable 5.1
Coding	62 tasks: 39 short Python implementations and 23 program-output predictions.	38 tasks: 20 SciCode and 18 LiveCodeBench (Jain et al., 2025).
Co-work	20 tasks: 12 numeric answers and 8 short format/constraint tasks.	42 tasks: 26 Advanced-IF and 16 GDPval professional tasks.
Reasoning	25 tasks: 3 short HLE multiple-choice questions and 22 short numeric/probability problems.	30 HLE tasks.
Response form	Short function, program output, scalar/fraction, option letter, or constrained note.	Scientific/competitive code, instruction following, professional tasks, and HLE, verified in the frozen suite.
Scoring weights	Nonuniform: six co-work tasks carry 93.87%; three HLE tasks carry 3.60% of reasoning.	Coding/reasoning equal-weighted. Co-work IF weight 1 and GDPval weight 1.5.

Table 5: **Task forms and weights at similar nominal scale.** Grok has 107 questions and Fable 5.1 has 110.

Candidate	Grok cost	Fable 5 cost	Cost ratio	Completed items
GPT-5.5	\$1.5869	\$26.55	16.73×	107 / 115
GPT-6 Astra	\$3.5767	\$25.47	7.12×	107 / 116

Table 6: **Candidate-matched generation expenditure.** Helper grading is excluded. Completed counts are Grok / Fable 5.

Score spread. GPT-5.5, Astra, and Sol all score 100% on Grok co-work, while Fable 5.1 reports 79.22%, 81.73%, and 70.52% for the same candidates. This contrast, alongside Grok's 74.87% overall PA and 22.59 overall Score, illustrates why task cost, score spread, and target agreement must be assessed separately.

Frozen execution allowances and grading paths. The supplied Fable 5.1, Fable 5, and Opus 5 checkpoints match every file hash in their respective freeze manifests. Table 7 summarizes their requested output allowances alongside Grok. The sum is a protocol setting for one full suite, excluding helpers and retries; it is not observed token consumption or a billing bound.

Artifact	Questions	Sum of caps (M tokens)	Per-question caps
Grok 4.7	107	0.115	768–2,048
Fable 5.1	110	1.374	5,000–16,000
Fable 5	118	2.261	16,384–24,576
Opus 5	108	1.968	8,000–24,000

Table 7: **Declared answer allowances at similar question counts.** A larger allowance permits longer answers and reasoning; it does not require every call to use that many tokens.

Fable 5.1 grades GDPval against upstream rubric criteria using `helper-grader`; its free-response reasoning uses local matching followed, when needed, by `helper-simulator`. The corresponding known assessment charges are \$33.9709 and \$0.0070. Fable 5 uses the same division of grader roles, with up to three rubric-judge attempts when a verdict cannot be parsed (\$18.0167 rubric and \$0.0038 equivalence charges). Opus uses deterministic structural/numeric checks for co-work and only the cheap reasoning equivalence path (\$0.0193). Grok has no assessment helper calls. These partitions reconcile to the framework's recorded evaluation totals; outstanding reservations remain separate at \$7.1444, \$1.9747, and \$7.6783 for the three supplied runs.

Observed decisions behind the allowances. Fable 5.1's early valid snapshots contain 145–153 items including ArXivMath (Dekoninck et al., 2026). Its pilot tool outputs record empty capped ArXivMath responses on both tested candidates. Later snapshots remove that family and reduce the suite to 110 items. A completed development pass then records nine capped GDPval answers

Questions	SD(Score)			SD(PA) (pp)		
	Astra	GLM	K3	Astra	GLM	K3
10	6.64	6.92	6.49	3.74	3.83	4.09
30	5.48	3.97	4.06	2.97	3.20	3.24
60	3.70	2.27	2.21	2.17	2.49	2.26
90	2.14	0.96	1.14	1.34	1.27	1.51

Table 8: **Metric variability across question subsets.** Overall-score standard deviations over 1,000 draws with stored responses and the twelve-candidate panel held fixed. Score is defined in Section 3.3; PA variation is in percentage points.

out of sixteen, prompting a final increase from 8,000 to 16,000 tokens for that family. Opus's development diagnostic records 36 capped responses out of 102, including 32 empty replies; its final code increases reasoning, coding, and co-work caps to 24,000, 16,000, and 8,000. These diagnostics establish why the researchers changed the settings, without assuming the larger caps necessarily improve every final metric.

Fable 5's trajectory shows that grading is also part of the iteration budget. Its rubric helper initially exhausts its allowance without a parsable verdict; the researcher increases that allowance, adds bounded retry samples, and changes error handling to preserve other item results. Slow candidate calls and late grading failures delay the response matrix needed for weight selection, so the final suite retains its predefined family weights. Monetary spend, feedback latency, and deployment cost are therefore distinct constraints on evaluation research.

B Frozen-Suite Sensitivity and Source-Stratified Baseline

Smaller question sets cost less to execute, but may change both agreement and measurement variability. To examine this trade-off, the question-count analysis below uses a fixed common panel of twelve candidates and fourteen targets.

B.1 Source-stratified Random120 baseline

Random120 tests how much agreement can be obtained by sampling existing public tasks without iterative research. We fix source-level quotas to cover multiple task types, then sample within each source independently of target reference scores. The maintainers selected sixteen sources, screened 21,083 eligible items, set source quotas, adapted fixed local graders, and used equal item weights for each raw domain mean. A Python pseudorandom generator with seed 42 sampled only *within* each source. Coding draws eight tasks each from MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021), DebugBench (Tian et al., 2024b), LiveCodeBench (Jain et al., 2025), and RepoBench (Liu et al., 2023b). Co-work draws five from IFEval (Zhou et al., 2023), six each from BFCL (Patil et al., 2025), ACPBench (Kokel et al., 2026), InFoBench (Qin et al., 2024), LIFBench (Wu et al., 2025), and PlanBench (Valmeekam et al., 2023), and five from BEIR (Thakur et al., 2021). Reasoning draws ten each from LogiQA (Liu et al., 2020), MMLU (Hendrycks et al., 2021), GPQA (Rein et al., 2023), and AIME25 (Balunovic et al., 2026). The resulting frozen suite has 40 tasks per domain, 120 in total. Twelve candidates completed all three domains, yielding 7–11 matched candidate scores per target across fourteen references. The main-table Random120 row uses these observed scores; the matched comparison recomputes all eight artifacts on this same candidate panel.

B.2 Matched-panel comparison and score decomposition

To compare the agent-designed evaluations with Random120 without changing the candidate panel, we recompute all eight complete-domain artifacts on exactly the same twelve candidates and fourteen target-reference intersections as Random120. Every target retains its original score scale and eligibility; no researcher program or output is changed. Table 9 gives all values and the alternative domain-equal aggregate.

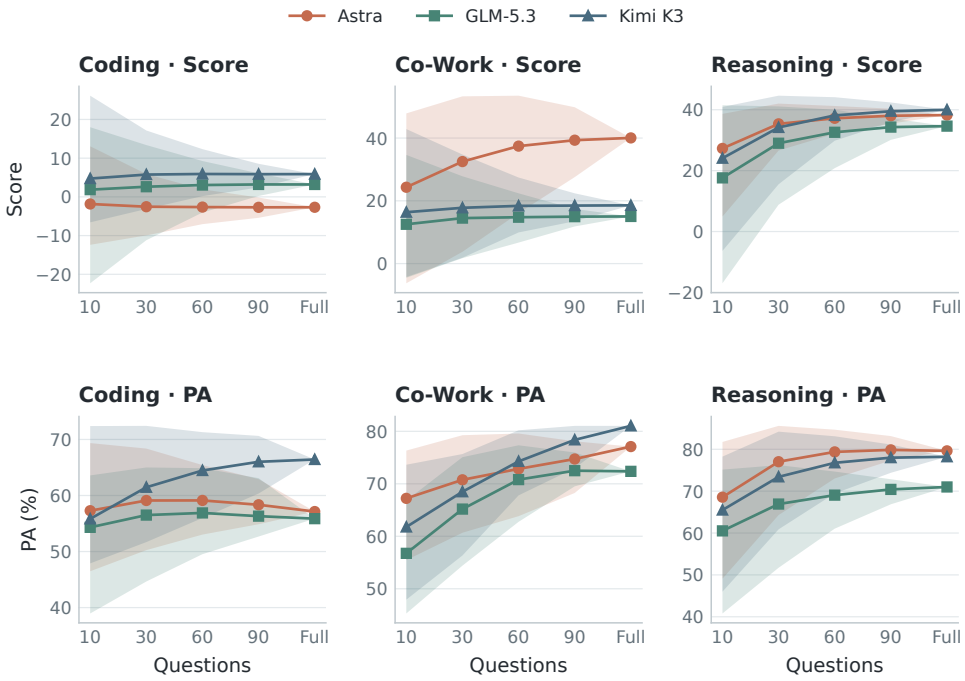

Figure 8: **Question-count sensitivity by domain.** Columns show coding, co-work, and reasoning; rows show Score and PA. Lines and bands give means and empirical 2.5–97.5 percentiles over 1,000 nested domain-stratified subsets on twelve common candidates. Full contains 112/100/107 questions for Astra/GLM/Kimi.

	Target-equal Score				PA	Domain-equal
Artifact	Coding	Co-work	Reasoning	Overall	Overall	Score
Astra	-2.69	40.05	38.25	21.35	69.08	25.20
Grok 4.7	22.94	25.75	14.71	22.18	76.07	21.14
Fable 5.1	12.79	19.89	36.87	20.48	68.46	23.18
Opus 5	-3.83	37.18	39.78	20.16	64.81	24.37
Kimi K3	5.88	18.55	39.99	17.72	74.17	21.47
Sol	19.29	0.17	41.06	17.13	62.55	20.17
GLM-5.3	3.20	15.02	34.62	14.15	65.01	17.61
DeepSeek V4 Pro	0.59	8.92	32.72	10.45	63.51	14.08
Random120	22.92	3.75	9.03	13.10	68.29	11.90

Table 9: **All artifacts and Random120 on the same twelve-candidate panel.** Every method uses the same candidate-reference intersections for each of fourteen targets. Target-equal Score gives each benchmark equal weight; domain-equal Score first averages targets within each domain, then gives each of three domains equal weight. PA is in percent.

For candidate-panel sensitivity, 10,000 multinomial draws (seed 20260925) resample twelve candidate identities with replacement. The same multiplicities apply jointly to all methods and all references, preserving within-candidate dependence. We recompute population moments and pairwise agreement, weighting candidate pairs by the product of their multiplicities. A common mask retains 9,916 draws with defined metrics for every method and target; the other 84 contain a target with no non-tied reference pair. The figure reports central percentile ranges of paired differences from Random120 on this fixed target set, not research-episode variation or uncertainty from selecting new target families.

For each target, with population standard deviations s_x, s_y and means $\bar{x}, \bar{y}$, CCC decomposes exactly as

$$\mathrm{CCC} = rC_b, \qquad C_b = \frac{2s_x s_y}{s_x^2 + s_y^2 + (\bar{x} - \bar{y})^2}.$$

Here r is score correlation and $C_b \in [0, 1]$ measures agreement in location and dispersion. With constant measurements we use $r = C_b = 0$, preserving CCC zero against a nonconstant reference. Table 10 reports mean per-target factors; their product is not used to approximate macro-CCC. The decomposition describes the frozen score geometry and performs no calibration or label-based fitting.

	Coding		Co-work		Reasoning		Overall Score	
Artifact	$\bar{r}$	$\overline{C_b}$	$\bar{r}$	$\overline{C_b}$	$\bar{r}$	$\overline{C_b}$	Target-eq.	Domain-eq.
Astra	0.079	0.508	0.521	0.808	0.708	0.505	21.35	25.20
Grok 4.7	0.511	0.393	0.432	0.667	0.591	0.302	22.18	21.14
Fable 5.1	0.321	0.498	0.328	0.614	0.633	0.527	20.48	23.18
Opus 5	0.103	0.510	0.451	0.817	0.695	0.524	20.16	24.37
Kimi K3	0.269	0.367	0.457	0.497	0.704	0.518	17.72	21.47
Sol	0.486	0.397	-0.097	0.427	0.701	0.507	17.13	20.17
GLM-5.3	0.191	0.405	0.514	0.275	0.585	0.517	14.15	17.61
DeepSeek V4 Pro	0.124	0.157	0.384	0.185	0.561	0.505	10.45	14.08
Random120	0.529	0.480	0.091	0.740	0.409	0.335	13.10	11.90

Table 10: **Association, calibration, and aggregation on twelve matched candidates.** Per-target CCC is rC_b; C_b penalizes mean and dispersion mismatch. Factor means are reported separately, so $\bar{r}\,\overline{C_b}$ is not macro-CCC. Overall Score weights either all fourteen targets equally or each domain equally after averaging its targets.

Equal weighting of targets and equal weighting of domains answer different aggregation questions because domains contain different numbers of targets. On this matched panel, target-equal weighting gives Grok Score 22.18 and Astra 21.35; domain-equal weighting gives Astra 25.20 and Grok 21.14. The main leaderboard retains its target-equal definition.

B.3 Question-count protocol

Astra, GLM-5.3, and K3 contribute frozen suites of 112, 100, and 107 questions. For each suite, we draw 1,000 nested, domain-stratified subsets of 10, 30, 60, and 90 questions and compare each with its full suite. The same subset is applied to every candidate; retained item weights are renormalized within each domain. Saved completed grades, including wrong and empty answers, are reused without new model calls. Metrics are recomputed per target before target-equal averaging. The percentile bands describe variation across question draws on this fixed panel.

At full size on the twelve-candidate panel, overall (Score, PA) is (21.35, 69.08%) for Astra, (14.15, 65.01%) for GLM, and (17.72, 74.17%) for K3. At 90 questions, 34.4%, 70.5%, and 63.3% of the respective draws fall within one Score point of their full-suite value. These percentages describe within-artifact question subsets, rather than the ordering of different researchers.

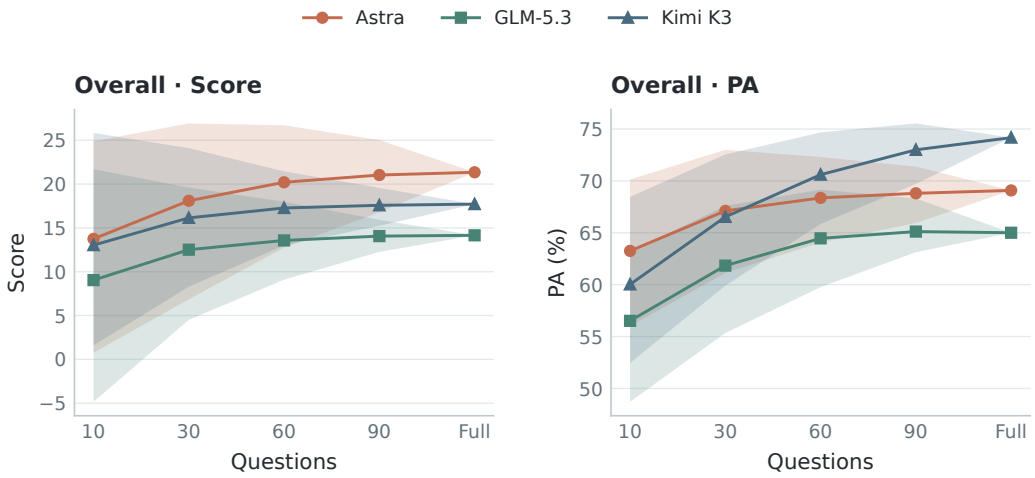

Figure 9: Overall Score and PA across nested question subsets. Bands show empirical 2.5–97.5 percentiles over 1,000 draws on the fixed candidate panel.

C Research Decisions and Native-Component Analysis

To determine which parts of a submitted suite contribute to its measurements, we group questions by the task families defined in that submission and remove each group in turn. This component audit reuses saved item responses for twelve candidates with complete scores across Astra, GLM-5.3, and K3, and the same fourteen target references. It uses the fixed score scales and target-equal aggregation in Appendix A.

For a task group (a *native component*) C, we remove its questions, renormalize the remaining frozen weights within that domain, and recompute Score and PA. The reported effect is the full Score minus the Score after deletion. Each effect is compared with 1,000 same-domain random deletions of the same number of questions. These offline comparisons reuse responses and do not rerun changed prompts.

Native grouping. Frozen task metadata define eleven Astra, six GLM, and nine K3 disjoint components. Their sizes and observed effects appear in Figure 11.

Execution effects. Removing Astra's eight spreadsheet tasks reduces co-work Score by 31.94 points and PA by 12.57 percentage points, while 46/96 responses in that component lack final answers at the output cap. Removing the ArXiv-math components (Dekoninck et al., 2026) raises reasoning Score and PA for all three suites; removing GLM's or K3's HLE choices lowers reasoning PA. Figure 10 shows termination counts for selected components.

The complete 26-component analysis appears in Figure 11; all completed wrong and empty grades remain unchanged.

C.1 Grok: from pilot response patterns to frozen weights

Grok first probes hard HLE and coding tasks, encounters truncated responses, and builds a bank of shorter, locally graded questions. The frozen suite contains 62 coding, 20 co-work, and 25 reasoning items. Its saved weight-selection program groups questions by their response patterns on three development candidates. For co-work, the GDPval references are close together. The program allocates most weight to jointly passed items and smaller masses to separating items, searching for the reference ordering and a slope close to one. This search constrains relative differences without matching the reference mean. Six co-work items receive 93.87% of total weight; the weight-

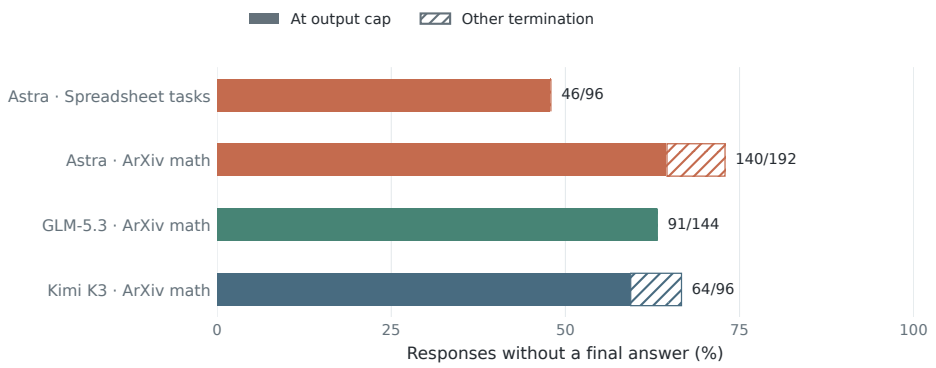

Figure 10: Missing final answers in selected components. Solid bars reach the output cap; hatched extensions indicate other termination reasons.

equivalent count $(\sum_i w_i)^2 / \sum_i w_i^2$ is 6.78. This quantity describes weight concentration, not the number of statistically independent questions.

The counterfactual uses the same saved grades and candidate–target intersections as the Random120 comparison. It replaces frozen weights with uniform within-domain weights, retaining completed wrong and empty answers and the existing infrastructure exclusions. Recomputed frozen aggregates reproduce all published matched-panel Grok values. Table 11 reports every domain, plus a sensitivity check restricted to the ten candidates with all items complete. The comparison changes aggregation only and makes no new model calls.

		Frozen weights		Uniform weights	
Candidate panel	Domain	Score	PA (%)	Score	PA (%)
Matched twelve	Coding	22.94	73.90	36.85	73.83
	Co-work	25.75	77.75	50.76	78.76
	Reasoning	14.71	77.62	14.53	75.76
	Overall	22.18	76.07	37.03	76.01
Complete ten	Coding	18.76	70.88	34.61	75.14
	Co-work	31.65	82.13	51.30	82.14
	Reasoning	11.07	83.36	12.67	83.81
	Overall	21.71	77.57	35.87	79.50

Table 11: **Grok aggregation counterfactual.** The matched panel uses the same twelve candidates as the Random120 comparison. The complete-item panel retains the ten candidates with all 107 grades available. Targets and within-panel item exclusions remain fixed between variants. These are offline counterfactuals; the primary leaderboard retains the submitted weights.

C.2 DeepSeek V4 Pro: grading convenience and measurement coverage

DeepSeek's source inspection identifies deterministic grading as a practical way to keep evaluation inexpensive. It constructs Python output-prediction, function-writing, and bug-fixing tasks, plus co-work checks for specified text, numbers, and formats. For example, one co-work question asks for a letter count but supplies the required output string; another asks for time conversion and supplies the converted time. Such checks measure compliance with an explicit response more directly than independent professional problem solving.

The trajectory includes genuine revisions: coding tasks are hardened, a domain-name mismatch in feedback analysis is repaired, and truncated reasoning pilots lead to larger answer allowances and a different question mix. The final reasoning bank includes short-answer HLE questions whose expected answers are easy to parse but whose mathematical content remains demanding. Selecting

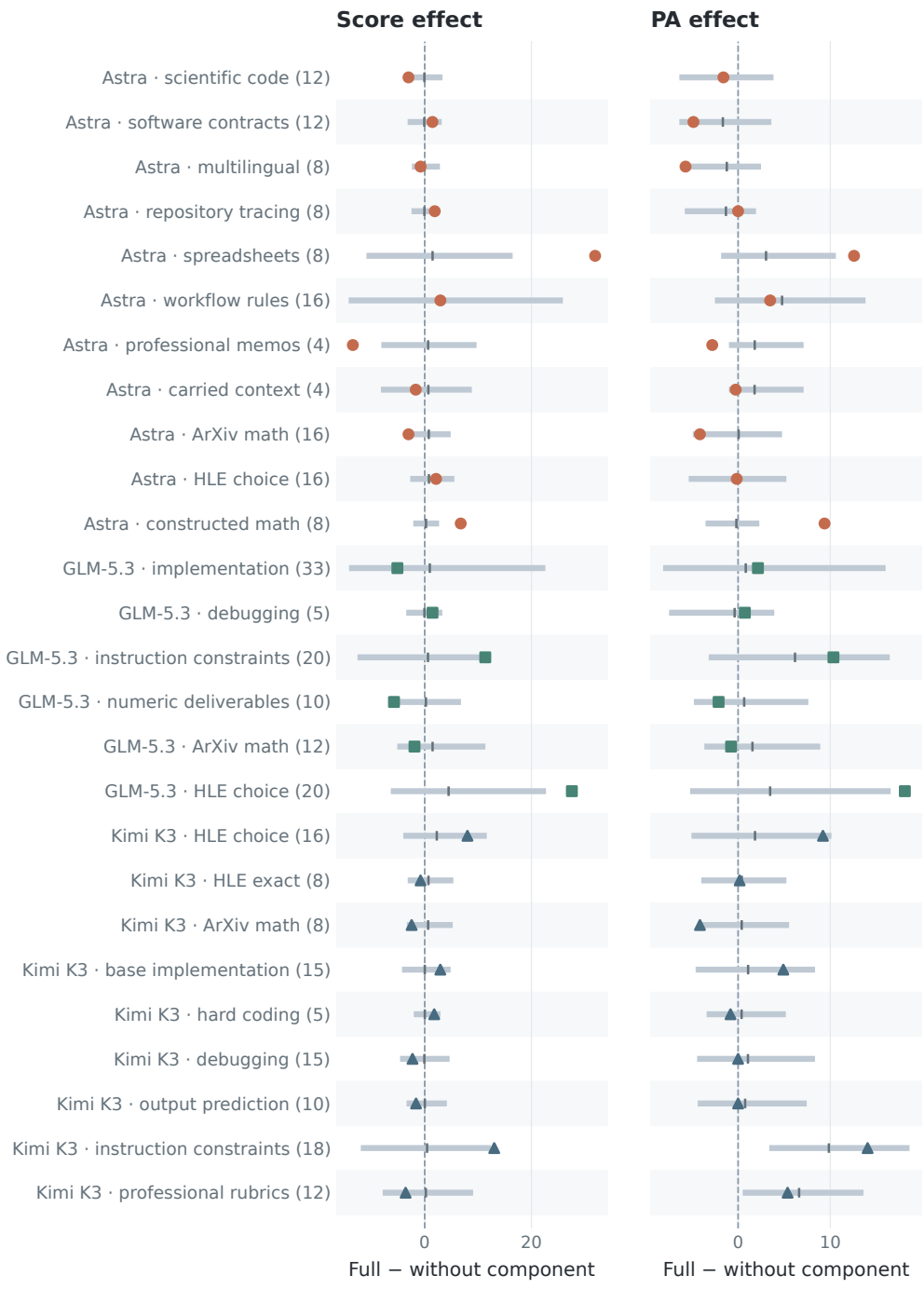

Figure 11: All 26 native components. Score effects are in Score points; PA effects are percentage points. Positive values mean removal lowers performance. Gray ranges show central 95% intervals from matched random removals.

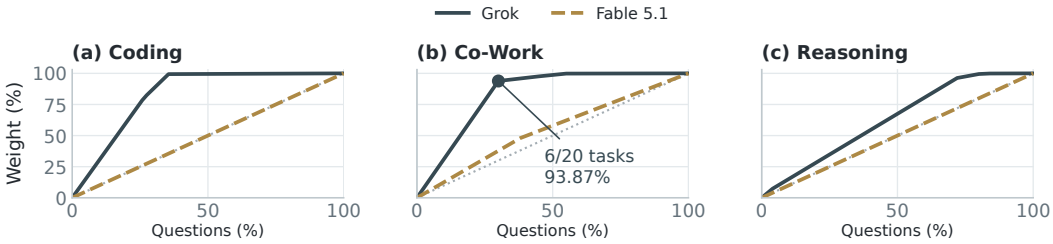

Figure 12: **Weight concentration across domains.** Cumulative scoring mass after sorting questions by decreasing weight, for Grok and Fable 5.1. The diagonal represents uniform weights.

a short answer simplifies grading, not necessarily reasoning. Final item counts are 38 coding, 51 co-work, and 30 reasoning, with equal within-domain weights.

On the matched panel, DeepSeek's coding and co-work means are high and its concordance calibration factors are low (Table 10). Its reasoning Score exceeds Random120's, whereas coding accounts for the largest deficit (Table 9). These observations support the interpretation that simplifying task form can preserve answer checkability while compressing the capability differences the proxy is intended to recover. They do not isolate task content from grading and execution as separate causal interventions.

D Research Instructions and Inputs

Researchers are asked to design one executable evaluation with at least 100 distinct scored questions spanning coding, co-work, and reasoning. They may select, adapt, or synthesize tasks and choose graders, weights, grouping, and output allowances. During development, researchers compare directly measured candidate-domain scores with visible target-reference scores; final assessment reports Score and PA for frozen programs.

The supplied workspace contains the task contract, development candidate APIs and available reference scores, original benchmark materials, available trajectories, and a Python evaluation SDK. The SDK submits a suite, records item-level responses and costs, and returns cross-model feedback on completed development tests. Researchers decide what to test, revise, and freeze within the three-hour research period, $100 researcher wallet, and $100 shared development/helper wallet. Frozen assessment has a separate $120 wallet and $30 per-candidate cap.

The final program supplies executable questions, grading rules, and three fixed domain aggregations. Completed wrong or empty answers retain their grades; infrastructure failures remain incomplete. After freezing, the executor runs the candidate panel without sending assessment results back to the researcher.

E Case Study: How Researchers Revise Evaluations

The aggregate analyses show which design choices affect measurement, but do not explain how agents arrive at them. We now follow the saved research trajectories from source inspection through pilot testing and revision. Astra provides a concrete example of this loop; GLM and K3 illustrate different responses to development feedback.

Astra turns source inspection into a coverage hypothesis. A compact coding evaluation can miss repository-level behavior if it concentrates on isolated implementations. After inspecting upstream repository-investigation tasks, Astra identifies a gap in its implementation-heavy coding bank and constructs eight multi-file repository traces. In one, candidates predict cached outputs and recomputation counts after in-place mutation versus replacing a dependency object. Running the supplied fixture generates the expected outputs for grading (Figure 13a).

Pilot feedback can expose defects in the evaluation itself. A low pilot score may expose an unclear question or an overly restrictive grader, as well as a candidate error. Astra examines these possibilities in a performance-plan memo that scores 11/12: its response reports that 3/13 orders finish on time, but omits the mean close time required by the rubric. Astra finds that the question never explicitly requested that statistic and adds it to the brief. A later audit also relaxes a cloud-migration rubric that prescribed one architecture despite an open-ended brief, retaining the stated security and delivery requirements (Figure 13b,c). These revisions align prompts and graders before freezing; the saved pilot score precedes the repairs.

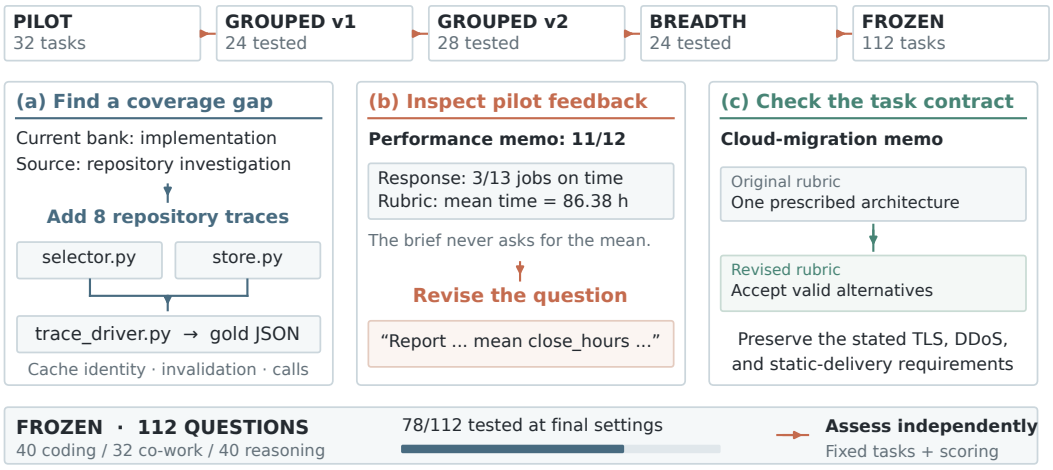

Figure 13: **Astra's research loop.** Source inspection motivates repository-tracing tasks; an 11/12 pilot exposes an implicit requirement, prompting a clearer question; a later validity audit broadens an overly restrictive rubric. The top strip shows saved versions (batch counts are not cumulative). The 112-question submission is then frozen for independent assessment.

Researchers make different revisions from similar feedback. GLM corrects invalid examples and numerical answers; K3 responds to near-ceiling pilots with five hard coding tasks. Astra also changes answer allowances and grouped execution after truncated responses. These examples show three uses of pilot feedback: correcting task validity, extending coverage, and changing execution settings. The component analysis in Section 4.3 tests whether the resulting task groups improve agreement on the frozen candidate panel.