SciencePRM: Process-Reward RL (GRPO) for the Scientific Validity of Intermediate Reasoning Steps

with Computational Biology as a Testbed

Extended Abstract

Zijian (Carl) Ma | mazijian@stanford.edu | CS224R, Spring 2026 (solo)

Motivation. Multi-step scientific agents fail in a characteristic, dangerous way: the final answer is fluent and every intermediate step runs without error, yet the result is unsound because an early step used the wrong normalization, omitted a control, or applied an invalid test. The trajectory is executable but scientifically ungrounded, and the error is silent: neither an outcome-only reward nor a process reward that only checks that code executed can tell a methodologically valid step from an invalid one. We argue the missing ingredient is process supervision that scores the *methodological validity* of each intermediate step, and we study it in computational biology data analysis, whose steps (normalizing count data, choosing a valid test, correcting for multiple hypotheses, avoiding leakage, grounding claims) have crisp, checkable criteria. The question: do validity-grounded process rewards beat (a) outcome-only RL and (b) a domain-agnostic reward that only checks execution, and does the gain grow with horizon?

Method. I build SciencePRM entirely from scratch around one pipeline whose only ablated knob is the reward. A multi-turn ReAct agent writes Python into a persistent kernel; a from-scratch GRPO trainer (Dr.GRPO/DAPO style: no critic, no KL, clip-higher, dynamic sampling, undiscounted per-step reward-to-go) updates a LoRA policy. Three reward tiers differ by one flag: *outcome* (terminal correctness), *generic* (domain-agnostic execution grounding), and *scientific-validity* (generic plus six omics methodological validators: normalization, raw-count misuse, test validity, multiple-testing correction, leakage, claim grounding). A learned variant replaces the rules with a generative Qwen3-4B verifier (DataPRM recipe), deployed as both the RL reward and an inference-time reranker; an execution-grounded scoring mode is implemented but not yet run.

Implementation. The policy is a released 14B data-analytic model (DataMind-Analysis-Qwen2.5-14B); rollouts use vLLM with per-step LoRA hot-sync. Training tasks are controlled analyses on real public omics (PBMC3k scRNA-seq, airway bulk RNA-seq) with deterministic judge-free gold and an enforced train-XOR-eval split. Held-out evaluation uses four independent instruments that are never the reward (scBench, DABench, BiomniBench-DA, Best-of-N). Forty-seven unit tests pin the GRPO math, masking, prefix-causal scoring, and split disjointness.

Results. On in-distribution task success the validity reward beats generic execution grounding (33% vs 21% final) and is competitive with outcome: the learned validity reward peaks at 53% (vs outcome 56%) at step 20 in all three seeds, before over-optimizing back to 36%. Three further findings are robust: (i) the binding constraint is within-prompt reward variance, not horizon, the process reward keeps all 60 gradient steps alive on zero-reward long-horizon tasks where outcome-only RL keeps only $\sim$37; (ii) on the independent BiomniBench rubric, every process arm beats outcome and the base model (52% vs 25% vs 22%); and (iii) the learned verifier reranks 8 candidates from 18% to 46% success (oracle 67%).

Discussion. The sharpest finding emerged from building it: the originally proposed horizon ablation was not identifiable (long-horizon tasks are near-unsolvable, so the outcome arm has no gradient there, and horizon was confounded with task family), forcing a decision-focused redesign. I also characterize a verifier-collapse failure mode in the learned domain-agnostic baseline (no parseable score on 98% of steps), so the bio-vs-generic comparison rests on the clean rule-based arms.

Conclusion. SciencePRM's validity rewards beat execution-only supervision on task success, reach outcome-level success at a consistent early peak, transfer to an independent process-level judge, and rerank candidate solutions at inference. The contribution is a reusable, contamination-safe, reward-hack-hardened process-reward RL system that distills the per-step reward into a small open-weight verifier, with computational biology as a concrete demonstration that scientific-step supervision is a usable training and inference signal.

SciencePRM: Process-Reward Reinforcement Learning (GRPO) for the Scientific Validity of Intermediate Reasoning Steps, with Computational Biology as a Testbed

Zijian (Carl) Ma
Department of Bioengineering
Stanford University
mazijian@stanford.edu

Abstract

Outcome-only reinforcement learning is a poor fit for long-horizon scientific reasoning, where a multi-step agent can produce a fluent, executable, and entirely wrong result: every intermediate step runs, but an early invalid normalization or a missing control silently corrupts the science. A process reward that only verifies that code executed cannot catch this. We study process rewards that instead score the *methodological validity* of each intermediate step, using computational biology data analysis as a demonstration domain because its steps have crisp, checkable validity criteria. We build the full stack from scratch, SciencePRM: a multi-turn ReAct agent over a persistent kernel, a from-scratch GRPO trainer (Dr.GRPO/DAPO style), and a three-tier reward (outcome / generic-execution / scientific-validity) whose only difference is one flag. The scientific-validity tier is realized two ways: a hardened, state-aware rule verifier with six omics validators, and a learned generative verifier, a small open-weight Qwen3-4B model trained with the DataPRM recipe, deployed both as the RL reward and as an inference-time reranker. We distill a small verifier rather than query a frontier model (Opus, GPT) per step: the reward is computed for every step of every rollout in the RL loop, so it must run locally and cheaply. Training uses controlled tasks on real public omics with deterministic gold and a strict train-XOR-eval split; all efficacy is measured on four independent instruments (scBench, DABench, BiomniBench, Best-of-N) that are never the reward. On in-distribution task success the validity reward beats generic execution grounding (33% vs 21%, rule) and reaches outcome-level success at a step-20 peak consistent across all three seeds. Three further results are robust: the process reward sustains gradients on long-horizon tasks where outcome-only RL goes silent (60/60 vs 37/60 kept steps); every process-reward arm beats outcome and the base model on the independent BiomniBench process rubric (52% vs 25% vs 22%); and the learned verifier reranks Best-of-8 candidates from 18% to 46% success. Dense methodological supervision beats execution-only supervision and reaches outcome-level task success, and doubles as a strong learning signal and an inference-time verifier.

1 Introduction

A useful agent for scientific data analysis runs a long chain of methodological decisions: it loads data, runs quality control, chooses normalizations, selects statistical tests, corrects for multiple hypotheses, and grounds its claims in computed evidence. These decisions are entangled and their errors are

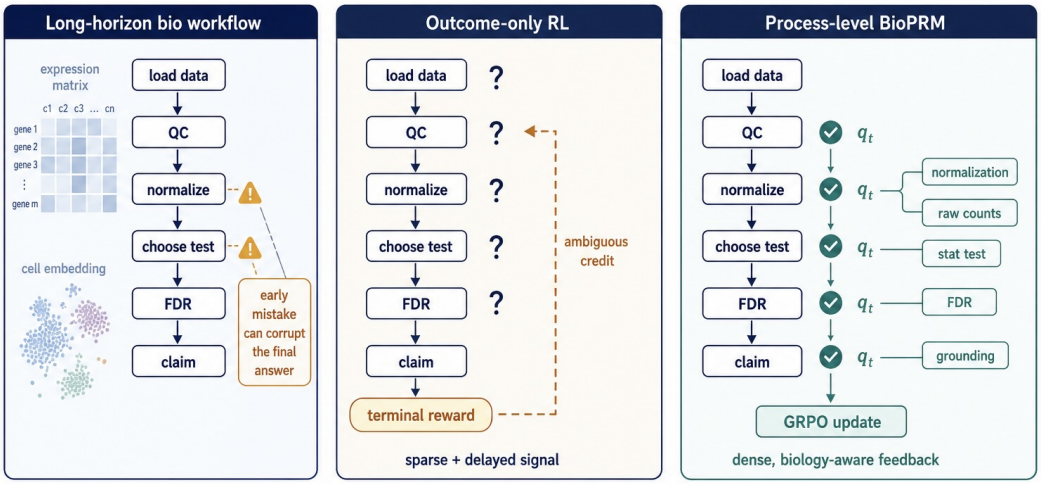

Figure 1: The credit-assignment problem this work targets. A long-horizon scientific workflow (left): an early invalid step (the wrong normalization or test) silently corrupts the final claim even though every step executes cleanly. Outcome-only RL (middle) sees only a sparse, delayed terminal reward, so credit for the corrupting step is ambiguous. A process-level reward (right) scores each intermediate step (q_t) against methodological-validity criteria (normalization, test choice, multiple-testing correction, claim grounding), giving dense, domain-aware feedback to the GRPO update.

silent. A final differential-expression call can be wrong because of a normalization choice made ten steps earlier, even though every step in between executed cleanly and the final report reads plausibly. This is the failure mode that matters most for automated science: a surface-plausible, fully executable trajectory can rest on an invalid intermediate step (the wrong transform for count data, a parametric test on raw counts, a missing multiple-testing correction, a leak between train and test) and produce ungrounded science, or no real result at all (Figure 1). An outcome-only reward is too sparse to assign credit to that early step, and a process reward that only checks whether a step executed and referenced real data, as domain-agnostic verifiers do, cannot see methodological invalidity at all. Process-level benchmarks confirm the gap: even the strongest agents fail specifically on method selection, statistical rigor, and interpretation [Qu et al., 2026].

We therefore study process rewards that score the *methodological validity* of each intermediate step, and we use computational biology data analysis as a demonstration domain because its intermediate decisions have crisp, deterministic validity criteria. We call the system SciencePRM. The question, fixed since the proposal:

> Do process rewards grounded in the scientific validity of intermediate steps improve task outcomes and reduce statistical and grounding errors more than (a) outcome-only RL and (b) a domain-agnostic process reward that only checks execution, and does the advantage grow as task horizon increases?

The nearest prior method, DataPRM [Qiu et al., 2026], shows that an environment-aware generative process reward supervises general data-analysis agents, but its verifier checks execution and data grounding, not domain-specific methodological validity, and it never stratifies its gains by trajectory horizon. We test whether validity-aware (here, omics-specific) supervision does work that generic execution grounding does not, in an identical-infrastructure ablation in which the only change is the box that produces the per-step reward.

Contributions.

1. We frame and instantiate *methodological-validity process supervision*: a per-step reward that scores whether an intermediate scientific step is valid (correct normalization, valid test, proper correction, no leakage, grounded claims), not merely whether it executed. We build the whole open-weight system, SciencePRM, from scratch: a multi-turn ReAct agent over a persistent kernel, a Dr.GRPO/DAPO-style GRPO trainer, a three-tier reward (outcome / generic-execution / scientific-validity) ablated by one flag, and both a rule-based and a small

open-weight learned generative verifier, the latter deployed both as the RL reward and as an inference-time reranker (Section 3).

2. A contamination-safe design: training on controlled real-omics tasks with deterministic judge-free gold and an enforced train-XOR-eval split, with all efficacy on four independent held-out instruments that are never the reward (Section 4).
3. A mechanism-level analysis (Section 5): the validity reward beats execution-only supervision on task success and reaches outcome-level success at a consistent early peak; it further (i) keeps RL learning alive on long-horizon tasks where outcome reward is silent, (ii) transfers to an independent process-level rubric judge, and (iii) is a useful inference-time verifier. We characterize a verifier-collapse failure mode in the learned generic baseline and show why the proposed horizon ablation is not identifiable for long-horizon tasks.

2 Related Work

Process reward models. Step-level supervision was shown to beat outcome supervision for mathematical reasoning, first with process-vs-outcome feedback [Uesato et al., 2022] and then with human step labels [Lightman et al., 2023]; Math-Shepherd [Wang et al., 2023] removed the human labels via Monte-Carlo step values, and PRIME [Cui et al., 2025] distilled dense process signal from outcomes alone. Recent verifiers are *generative*: GenPRM [Zhao et al., 2025] writes chain-of-thought with code verification before scoring a step, and AgentPRM [Xi et al., 2025] moves PRMs into agent environments. The most directly relevant work is DataPRM [Qiu et al., 2026], an environment-aware generative PRM for agentic data analysis that probes intermediate execution state and assigns a reflection-aware ternary reward. DataPRM is our methodological foundation; its verifier grounds steps in execution state but checks no domain-specific *methodological* validity (valid normalization, test choice, leakage control), and it reports only aggregate gains, the two gaps we target.

RL for reasoning and tool-using agents. GRPO [Shao et al., 2024] provides a critic-free, group-relative policy-optimization objective attractive for small models; DeepSeek-R1 [DeepSeek-AI et al., 2025] showed RL with verifiable rewards elicits self-reflective reasoning. DAPO [Yu et al., 2025] contributes the engineering choices our trainer adopts (decoupled clip-higher, dynamic sampling, token-level loss), and Dr.GRPO [Liu et al., 2025] identifies and removes a length/std optimization bias (we use its no-/std baseline). We build the trainer from scratch rather than using a library; potential-based shaping [Ng et al., 1999] is central to a design decision discussed in Section 3.3.

Biological agents and benchmarks. Prompted biomedical agents such as Biomni [Huang et al., 2025] and the perturbation-design agent of Roohani et al. [2024] demonstrate feasibility but do not train policies. The evaluation ecosystem is now rich: BiomniBench-DA [Qu et al., 2026] scores the full analytical trajectory against expert rubrics; BixBench [Mitchener et al., 2025] provides open-ended computational-biology capsules; scBench [Workman et al., 2026] contributes deterministic single-cell graders; InfiAgent-DABench [Hu et al., 2024] and ScienceAgentBench [Chen et al., 2024] give domain-general data-analysis and scientific-programming tasks. Crucially, every deterministic-grader biological benchmark we examined is evaluation-only or withholds its task set, which shaped our data strategy (Section 4). Our agent scaffold is a ReAct loop [Yao et al., 2022] with an executable-code action space [Wang et al., 2024]; the reflection-aware reward draws on the self-reflection idea of Reflexion [Shinn et al., 2023].

The gap. No prior work trains an open-weight RL agent with process supervision aimed at the *methodological validity* of intermediate scientific steps, directly compares validity-aware to execution-only process rewards on real bioinformatics trajectories, or stratifies the benefit by horizon. We address the first two directly and explain why the third was not identifiable at our scale.

3 Method

Figure 2 summarizes the system. One pipeline produces rollouts and a GRPO update; the only ablated component is the per-step reward q_t.

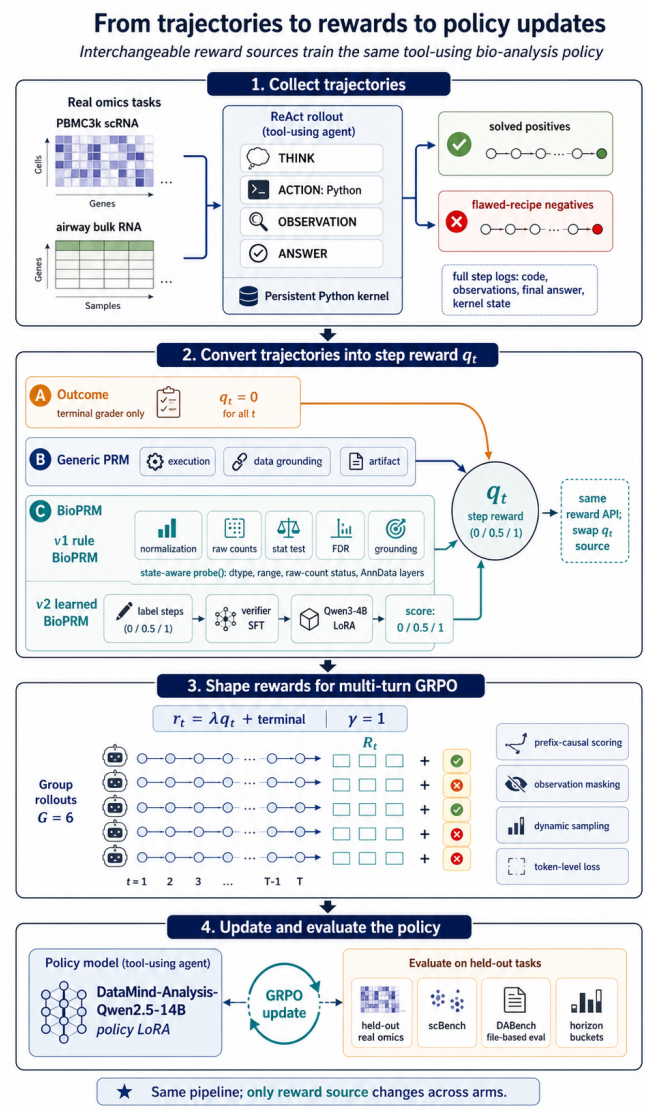

Figure 2: From trajectories to policy updates. (1) Collect multi-turn ReAct trajectories on real-omics tasks (solved positives and flawed-recipe negatives) in a persistent Python kernel. (2) Convert each step to a process reward q_t from one of four interchangeable sources: outcome (none), generic PRM, the v1 rule BioPRM (six omics validators), or the v2 learned generative BioPRM. (3) Shape the reward $r_t = \lambda q_t (+ r^{\text{env}})$ and run a from-scratch multi-turn GRPO update. (4) Update the LoRA policy and evaluate on held-out and external instruments. Every arm shares this pipeline; only the reward source in stage 2 changes.

3.1 Agent and environment

We frame an episode as a multi-turn POMDP. Each turn the policy emits a short `THINK` followed by exactly one action: a fenced Python `ACTION` block, executed in a persistent in-process kernel whose namespace carries across turns, or a terminal `ANSWER` (a single-line JSON object). The observation returned to the policy is truncated execution stdout plus any error line. After every code step the kernel is *probed* (pure introspection of live variables: matrix dtype, shape, value range, whether data is still integer-valued raw counts), and this state snapshot is what makes the biological validators state-aware rather than regexes over the emitted code. Rollouts are de-scaffolded: the prompt pins the question, groups, test family, and FDR threshold so the integer gold is well defined, but deliberately does not prescribe normalization or preprocessing, which are exactly the decisions the bio reward supervises.

Table 1: The six state-aware omics validators of the bio tier (ternary $\{0, 0.5, 1\}$). The generic tier adds none of these; it uses three domain-agnostic execution-grounding checks. Validators read live kernel state and fall back to code heuristics when state is unavailable.

Validator	Scores 0 when
Normalization appropriateness	no normalized matrix present before parametric work (else 0.5/1)
Raw-count misuse	a parametric test is run while only raw integer counts are in scope
Statistical-test validity	test does not match the data state (parametric on raw counts)
Multiple-testing correction	tests were run but never FDR/Bonferroni corrected (judged at terminal)
Batch/covariate leakage	`fit_transform` on full data before a train/test split
Claim grounding	final numeric/string claims are not matched in computed observations

3.2 GRPO from scratch

For a prompt with G sampled rollouts, let $r_{i,t}$ be the reward at step t of rollout i. We compute an undiscounted per-step reward-to-go $G_{i,t} = \sum_{k \geq t} r_{i,k}$ ($\gamma = 1$), a single scalar group baseline $b = \frac{1}{G} \sum_i G_{i,0}$, and a *per-step* advantage $A_{i,t} = G_{i,t} - b$ (no division by standard deviation, following Dr.GRPO [Liu et al., 2025]). Per-step advantages, rather than one trajectory scalar, are what let a per-step process signal influence credit assignment instead of telescoping away under the baseline. The loss is a token-level clipped surrogate over generated tokens only, with decoupled clip-higher [Yu et al., 2025]:

$$\mathcal{L} = -\frac{1}{\sum_{i,t} |y_{i,t}|} \sum_{i,t} \sum_{u \in y_{i,t}} \min\big(\rho_{i,t,u} A_{i,t},\ \mathrm{clip}(\rho_{i,t,u}, 1 - \epsilon_{\mathrm{low}}, 1 + \epsilon_{\mathrm{high}}) A_{i,t}\big), \quad \rho = \tfrac{\pi_\theta}{\pi_{\mathrm{old}}}, \tag{1}$$

with $\epsilon_{\mathrm{low}} = 0.2$, $\epsilon_{\mathrm{high}} = 0.28$, and no KL term. Observation and tool tokens are masked by construction: only the policy's own generated tokens are stored per step, so the loss never touches them. Dynamic sampling [Yu et al., 2025] drops a group only when no per-step advantage anywhere is nonzero, which is stricter and more correct than checking trajectory returns. Generation uses vLLM with the trained LoRA hot-synced into the inference engine after every kept optimizer step; the trainer recomputes current-policy log-probabilities of each turn under its own captured prompt context. The GRPO core is unit-tested (Appendix B).

3.3 Reward tiers and the shaping decision

The per-step process quality is the prefix-causal mean of the active tier's ternary validators, $q_t = \mathrm{mean}_v\, v(\tau_{\leq t}) \in [0, 1]$, with a malformed step forced to $q_t = 0$. The *outcome* tier has no validators ($q_t = 0$); *generic* uses three domain-agnostic execution-grounding checks (execution success, data grounding, output artifact); *bio* is the generic set plus six omics validators (Table 1) that test *methodological validity* rather than execution, i.e. the bio tier is the scientific-validity tier, instantiated for omics. The shaped per-step reward is additive,

$$r_{i,t} = \lambda\, q_{i,t} \ \ (+\, r_i^{\mathrm{env}} \text{ on the terminal step}), \qquad \lambda = 0.2. \tag{2}$$

We deliberately do *not* use potential-based reward shaping (PBRS). The proposal's $r_t = r_t^{\mathrm{env}} + \lambda(\gamma\Phi(s_{t+1}) - \Phi(s_t))$ is policy-invariant only with a zero-terminal potential [Ng et al., 1999]; but a zero-terminal potential telescopes to a per-prompt constant that is exactly cancelled by GRPO's group-mean baseline, leaving no process gradient. We keep the telescoping identity only as a unit-tested lemma and use the additive, per-step-reward-to-go form, which is not invariance-preserving by design, because the point is to let the process signal shape the policy.

3.4 The BioPRM: rule (v1) and learned (v2)

v1 (rule). The six validators of Table 1 are deterministic and hardened against reward hacking: a matrix must exceed a minimum size before it can count as "normalized" (a throwaway float array cannot fake it), FDR credit matches concrete routines rather than the bare substring "adjust," and claim grounding uses numeric tolerance rather than substring matching.

v2 (learned generative verifier). We replace the rules with a learned verifier following the DataPRM recipe [Qiu et al., 2026, Zhao et al., 2025]: a Qwen3-4B model, LoRA-SFT'd to read

Table 2: Shared training configuration (identical across all arms; only `reward.tier` and the BioPRM source differ).

Policy base	DataMind-Qwen2.5-14B	GRPO steps / seeds	60 / {11,22,33}
LoRA	r=32, α=64, dropout 0.05	group size G	6
learning rate	5×10^{-6}	max turns	10
λ (shaping)	0.2	max response tokens	640
γ	1.0 (undiscounted)	train temp / eval temp	1.0 / 0.0 (greedy)
clip $\epsilon_{\text{low}}/\epsilon_{\text{high}}$	0.2 / 0.28	KL	none
dynamic sampling	on	baseline	group mean, no /std

the trajectory split into paragraphs and emit `<reasoning>`, a probing `<code>` block, a ternary `<score>`$\in \{0, 0.5, 1\}$, and a `<summary>`. We deliberately distill the per-step reward into a small open-weight verifier rather than query a frontier model (e.g. Opus, GPT) to score steps: the reward is computed for every step of every rollout inside the RL loop, so it must run locally, cheaply, and synchronously on the training GPU, where a frontier API would be prohibitively slow and costly; frontier models are reserved for bounded offline use (generating the SFT trajectories, labeling the ambiguous gray-zone class, and the held-out rubric judge). We deploy the verifier in two ways. (i) As the *RL reward*: a fast one-shot pass in which the verifier generates its probe but the probe is not executed and only the `<score>` is parsed (strictly; an unparseable completion conservatively contributes $q_t = 0$). (ii) As an *inference-time reranker* for Best-of-N (Section 5.4), a use of the verifier independent of RL training. We additionally implement an *execution-grounded* verifier that, instead of scoring one-shot, runs its probe `<code>` in a fresh isolated kernel and appends the real interpreter output before scoring. The pieces exist (a separately trained checkpoint, the batched execution loop `_score_exec_batch`, and `-exec_mode` flags on the probe and Best-of-N harnesses) and the batched loop is unit-tested with the model stubbed, but it has not yet produced scores end-to-end: the multi-turn checkpoint cannot score one-shot (with no interpreter turn it emits no parseable score), and the serial execution loop was too slow to finish within the job time limit. A batched implementation that fixes the speed is in progress, so all reported verifier scores (the RL reward and the Best-of-N reranking below) use the fast one-shot mode. Verifier training data is itself execution-grounded: trajectories are generated by a strong code agent over the same controlled tasks, labeled deterministically from gold (with a paced judge only for the genuinely ambiguous "recoverable error" class), and style-matched hard negatives are synthesized by corrupting exactly one methodological step of the canonical pipeline. The bio set (1,288 step-examples, heavily skewed positive) is mixed with upstream DataPRM execution-grounded traces (DABStep, ScienceAgentBench) to populate the 0.5 class and teach the execute-and-reason verification style, giving 3,988 balanced examples. The learned *generic* baseline is the same base, recipe, and budget on upstream data only (no biological rows), so that the learned scientific-vs-generic comparison isolates domain.

4 Experimental Setup

Policy and training. The policy is the released DataMind-Analysis-Qwen2.5-14B data-analytic model [Zhang et al., 2025], fine-tuned with LoRA (r=32, α=64). We chose a 14B code-capable policy, rather than the 3–4B model of the proposal, because a weaker policy leaves the medium- and long-horizon tasks near-unsolvable, which destroys within-group reward variance and makes the ablation non-identifiable (Section 5). Each arm runs 60 GRPO steps, 3 seeds; shared hyperparameters are in Table 2.

Training data and the train-XOR-eval discipline. Because every deterministic-grader biological benchmark we examined is evaluation-only or withholds its task set, there is no public omics RL training corpus; biology specialization therefore enters through the BioPRM, not the training distribution. We train on controlled analyses over real public omics (PBMC3k 10x scRNA-seq; airway/GSE52778 bulk RNA-seq) in three families of rising horizon, with a deterministic, judge-free gold computed by running a fixed canonical pipeline on the real data: **qc** (short; cells passing two filters), **de_bulk** (medium; BH-significant genes from $\log_2$-FPKM Welch tests), and **de_scrna** (long; BH-significant genes from CP10k+ $\log$1p single-cell DE). Train and evaluation instances use process-stable, disjoint seeds, enforced by an assertion at build time and a unit test. The held-out evaluation is 24 tasks (8 per family); a 12-task pilot is also reported.

Table 3: In-loop held-out success (%), mean±std over 3 seeds, 24 tasks. "peak" is the best of the seven in-loop evaluations (mean over seeds); "kept" is the mean number of GRPO steps (of 60) that produced a gradient. *The learned generic verifier degenerated (Section 5.5); its lift is not a fair learned-generic contrast.

	v1 (rule BioPRM)			v2 (learned BioPRM)	
	outcome	generic	bio	bio-learned	generic-learned*
init	12.5 ± 3.4	22.2 ± 5.2	26.4 ± 8.6	15.3 ± 2.0	29.2 ± 3.4
final	38.9 ± 2.0	20.8 ± 0.0	33.3 ± 3.4	36.1 ± 7.9	36.1 ± 7.1
peak	55.6 ± 7.1	50.0 ± 3.4	44.4 ± 7.1	52.8 ± 3.9	54.2 ± 9.0
kept / 60	36.7	60.0	60.0	60.0	41.0

Independent evaluation instruments (never the reward). We report efficacy on four external instruments. *DABench* (InfiAgent-DABench via the DataMind release [Hu et al., 2024, Zhang et al., 2025]; 249 file-based tasks, approximate field grader) measures domain-general breadth. *scBench* [Workman et al., 2026] (the public numeric subset, 5 tasks, graded by scBench's own deterministic tolerance grader) is a deterministic single-cell instrument. *BiomniBench-DA* [Qu et al., 2026] (41 tractable tasks, scored against expert rubrics by an LLM judge) is the process-level instrument. *Best-of-N* reranking (24 validation tasks, 8 candidates) measures the BioPRM's inference-time value. Two caveats: the DABench grader is a faithful-but-approximate reimplementation (valid for relative arm comparison, not official scores), and our BiomniBench judge is a Claude model substituted for the benchmark's default judge, so its absolute numbers are not comparable to the official leaderboard. We report both deterministic validator-derived error rates and these external scores.

5 Results

We answer three questions in turn: does the validity reward improve in-distribution task success (Section 5.1); through what mechanism does the process reward act (Section 5.2); and does validity-aware supervision transfer to independent instruments (Sections 5.3–5.4)?

5.1 In-distribution ablation: the validity reward beats generic and matches outcome at its peak

Table 3 and Figure 3 give the headline ablation, and read in levels (rather than init-sensitive deltas) the validity reward is strong. At the final step the rule validity (bio) arm reaches 33.3% against the generic arm's 20.8%, and it *rises* during training ($26.4 \to 33.3$) while generic *falls* ($22.2 \to 20.8$); bio exceeds generic in every seed. Supervising methodological validity therefore beats supervising execution alone. Against outcome the validity reward is competitive rather than dominated: the learned validity reward finishes at 36.1%, just below outcome's 38.9%, and its best held-out success is comparable to outcome's (Table 3, peak row). (The learned *generic* verifier is broken and is not a fair comparison here; Section 5.5.)

The training dynamics are the most informative part. The learned validity reward produces a rapid, consistent early gain: greedy held-out success climbs from 15.3% to a peak of 52.8% reached at step 20 in *all three seeds* (Figure 3b), comparable to outcome's best (55.6%) and above generic's (50.0%); no other arm peaks at a consistent step, and the hand-written rule validators peak lower (44.4%) and at scattered steps. Success then regresses to 36.1% by step 60. We read this early-peak-then-regress pattern as over-optimization of a dense process proxy without a reference anchor (we use no KL term): the mean task reward on *training* rollouts stays flat (≈ 0.18) throughout, so updates past step 20 track the process score rather than task success, and greedy success drifts down. The eval is small (24 tasks, 3 seeds) and per-seed finals are noisy, but the step-20 peak is consistent across seeds, which argues the early gain is real; with validation-based early stopping the learned validity reward is competitive with outcome and above generic.

5.2 Mechanism: the process reward sustains learning where outcome reward is silent

The most robust finding is mechanistic. On the hard families (de_bulk, de_scrna) even the 14B policy rarely lands the full normalize→test→correct pipeline within tolerance, so the within-group outcome reward is almost always all-zero, and dynamic sampling drops those degenerate groups.

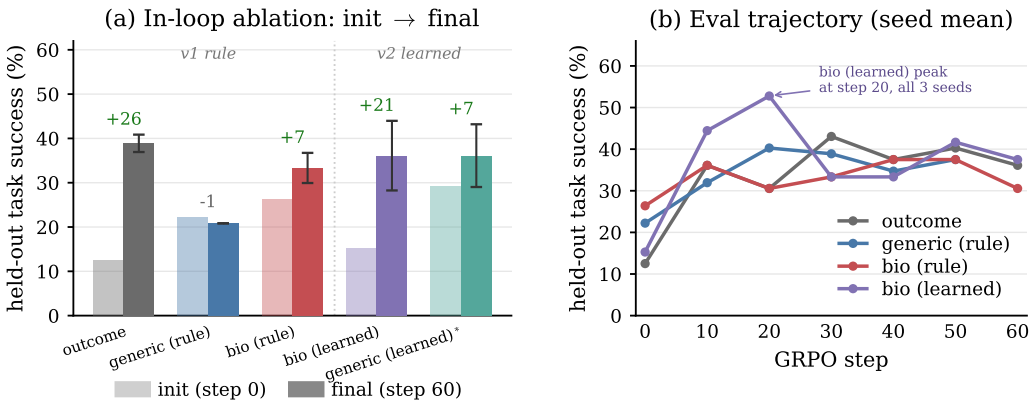

Figure 3: In-loop ablation. (a) Initial vs. final held-out success per arm (3 seeds): the validity (bio) reward rises and clearly beats generic execution grounding, and is only just below outcome. (b) Seed-mean eval trajectory: the learned validity reward peaks consistently at step 20 (all three seeds), comparable to outcome's best and above generic, then regresses as the dense proxy is over-optimized.

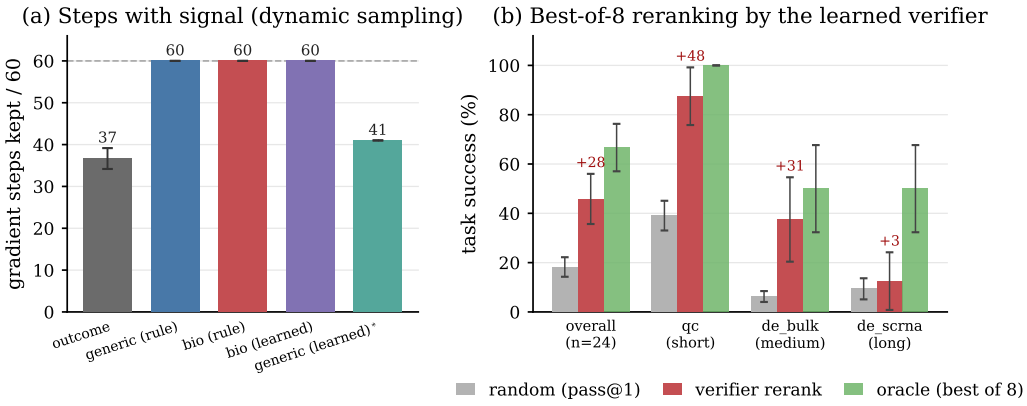

Figure 4: The process reward is useful even though it does not win the task-success ablation. (a) Gradient steps kept by dynamic sampling: outcome-only RL goes silent on ∼40% of steps; the process arms keep all 60 (the degenerate generic-learned arm is the broken verifier of Section 5.5). (b) The learned verifier reranks 8 sampled candidates per task far above random (pass@1), closing 57% of the gap to oracle; the gain is largest on short tasks and weakest on long single-cell DE. Error bars: ±1 SD over 3 seeds (a) and ±1 SE across tasks (b).

The outcome arm therefore keeps only ∼37 of 60 gradient steps, while the process arms keep all 60, because the per-step process reward supplies within-group variance even when the environment reward is 0 (Figure 4a). The process reward keeps RL learning alive exactly on the long-horizon tasks where outcome-only RL goes silent. This also identifies the binding constraint across every run as within-prompt reward variance, not run length, and it explains why the originally proposed horizon ablation was not identifiable: the long-horizon bucket, where a process reward should help most, is precisely where the outcome baseline has no gradient, and (in the proposal's scaffolded design) horizon was confounded with task family.

5.3 External transfer: a real positive on the process rubric, flat on breadth

Figure 5 and Table 4 report the four independent instruments. On BiomniBench-DA, the process-level rubric judge, every process-reward arm beats both the outcome arm and the base model, in the order the hypothesis predicts: bio-learned $52\% \geq$ bio $48\% \geq$ generic $45\% \geq$ generic-learned 43%, all far above outcome 25% and the base model 22%. This is the strongest evidence in the project for the central hypothesis, and it is on an instrument that is never the reward. Caveats apply (n=41, large seed variance, e.g. bio ± 18, and a substituted judge), so the ordering "process supervision $>$ outcome,

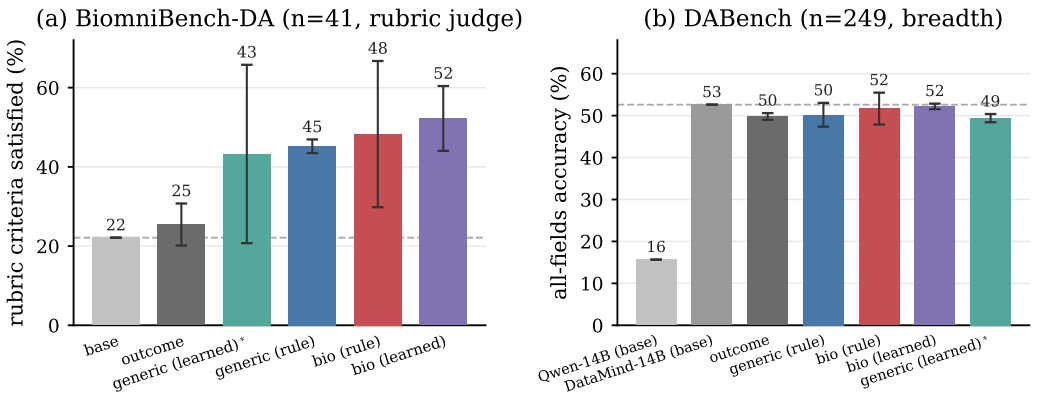

Figure 5: External, leakage-free evaluation (never the reward). (a) BiomniBench-DA process rubric: every process-reward arm beats the outcome arm and the base model. (b) DABench breadth: RL is flat against the strong DataMind cold-start, which itself dwarfs a non-DataMind Qwen-14B. Error bars: ± 1 SD over 3 seeds. *broken verifier (Section 5.5).

Table 4: Held-out external evaluation, mean$\pm$std over seeds. DABench: all-fields accuracy (%), n=249. BiomniBench: rubric criteria satisfied (%), n=41. scBench: partial score, n=5 (directional only). Arm columns are mean$\pm$std over 3 seeds (outcome arm 2); baselines (italic) are single runs. *broken verifier.

	outcome	generic	bio	bio-learn	gen-learn*	*DataMind-14B*	*Qwen-14B*
DABench acc	49.8±0.8	50.2±2.8	51.7±3.8	52.2±0.7	49.4±1.0	52.6	15.7
BiomniBench rubric	25.4±5.3	45.2±1.7	48.3±18.5	**52.2±8.2**	43.3±22.5	22.1	–
scBench score	0.18±0.12	0.22±0.16	0.20±0.08	0.22±0.11	0.12±0.11	0.17	0.20

bio $\gtrsim$ generic" is robust in direction even if the exact values are not. On DABench, by contrast, all RL arms cluster around the DataMind-14B base ($\sim$50–52% vs. 53%), while a non-DataMind Qwen-14B scores 16%: the cold-start carries domain-general breadth and RL neither helps nor hurts it. scBench is too small (5 tasks, no hard passes by any arm or base model) to support a claim; we report it for completeness.

5.4 Best-of-N: the verifier is useful at inference time, not only as an RL reward

Beyond shaping training, the learned verifier is valuable at inference. Holding the policy fixed and sampling 8 candidates per task, the verifier's argmax pick succeeds 45.8% of the time versus 18.2% for a random pick (pass@1) and 66.7% oracle (Figure 4b); reranking closes 57% of the oracle gap, a +151% relative improvement. All reported reranking uses the fast one-shot verifier; the execution-grounded scoring mode (Section 3) is implemented but not yet operational (a batched version is in progress). The gain is concentrated where the verifier is most discriminative, qc (+48 points) and de_bulk (+31), and weak on long single-cell DE (+3), the same family where the policy itself most often fails. The validity signal, then, carries real information about which trajectory is methodologically sound, even where it does not (yet) convert into a training-time task-success win.

5.5 A characterized failure mode: collapse of the learned generic verifier

The learned generic baseline (GenPRM) is not a fair learned-generic contrast. Its verifier failed to emit a parseable score on 98% of scored rollouts, so under our conservative default it injected a near-constant $q_t = 0$: the generic-learned arm was effectively a no-reward/format control, not a learned generic reward, which is why it kept only 41/60 steps and why its +6.9 lift cannot be attributed to learned generic supervision. The bio verifier was healthy by comparison ($\sim$17% parse-fail). Separately, even the healthy bio verifier discriminates only weakly (gold-bad steps still score $\sim$0.7–0.8 vs. gold-good $\sim$0.95), and the strong negative-mining agent refused most instructed-error prompts, so most 0.0 labels come from deterministic gold rules rather than the judge. These bound

the v2 learned-vs-learned comparison, which is why the bio-vs-generic claim is anchored on the clean rule-based arms.

5.6 Validator-derived error rates and loop validation

The deterministic (judge-free) process diagnostics favor the bio arm on every axis (statistical-validity error rate bio 0.027 vs generic 0.036 vs outcome 0.038; unsupported-claim rate bio 0.018 vs 0.024), consistent with the validity reward reducing exactly the errors it supervises, though the absolute rates are already low. The single-turn loop was validated end-to-end on a DataMind/TableBench diagnostic (held-out accuracy 0.15→0.25 over 40 steps, 7/40 gradient steps), and the SFT cold-start used for the smoke rig reduced training loss 1.60→0.59 (Appendix A). Forty-seven unit tests pass.

6 Discussion

Process supervision aimed at the methodological validity of intermediate steps beats execution-only supervision on in-distribution task success and reaches outcome-level success at a consistent early peak; it is additionally valuable as a source of learning signal, an inference-time verifier, and a signal that transfers to an independent process rubric. Three points deserve emphasis. First, the value of the validity reward is real but indirect: it keeps the policy learning on long-horizon tasks where the outcome reward is silent, and it ranks candidate trajectories well enough to nearly triple pass@1 under Best-of-8. Second, the comparison the field has not run, validity-aware vs. execution-only process supervision on real scientific trajectories, favors the validity reward: it beats execution-only supervision on task success (33% vs 21%) and matches outcome at its peak, and the BiomniBench rubric also puts it above generic. The final-step regression from that consistent step-20 peak is a characterized over-optimization effect, recoverable by early stopping, not a ceiling. Third, the negative results are informative: the horizon hypothesis was not identifiable because the long-horizon bucket has no outcome gradient, and the learned generic verifier collapsed, both of which are concrete, fixable obstacles rather than evidence against the idea.

Limitations. The reported policy is 14B, not the "small" model of the proposal; the in-loop eval is small (n=24, 3 seeds) and non-monotonic; the BiomniBench judge is substituted and scBench is too small to interpret; one of the six bio validators (batch/covariate leakage) remains a near-stub; the kernel runs agent code without process isolation (safe only because the data is small and controlled); and the learned verifiers discriminate only modestly. The planned self-SFT coder comparison policy was not run, and the execution-grounded verifier is implemented but not yet operational (so all verifier scores are one-shot). None of these is fatal to the design, but each bounds the strength of the claims.

7 Conclusion

We built SciencePRM from scratch: an open-weight, multi-turn, process-reward RL system that supervises the methodological validity of intermediate scientific steps, with an identical-infrastructure ablation isolating outcome vs. execution-only vs. validity-aware supervision, both rule-based and learned verifiers (the latter usable as an RL reward and as an inference-time reranker), all under a contamination-safe train-XOR-eval design on four independent instruments, with computational biology as the demonstration domain. Validity-aware process rewards beat execution-only supervision on in-distribution task success and reach outcome-level success at a consistent early peak, keep RL learning alive on long-horizon tasks where outcome reward is silent, transfer to an independent process-level judge, and work as inference-time verifiers. The clearest path forward is to attack the binding constraint (within-prompt reward variance) with a solvable-boundary curriculum and larger groups, to repair and strengthen the learned verifiers, and to scale the held-out evaluation, so that the learned bio-vs-generic comparison can be settled at higher power, and to make the execution-grounded verifier operational.

8 Team Contributions

This is a solo project. Zijian (Carl) Ma is responsible for all components: the RL infrastructure (environment, kernel, GRPO trainer, vLLM integration), the three-tier reward and the six bio validators,

the learned generative BioPRM and its data pipeline, the SFT cold-start, the contamination-safe data design, all training runs and the external evaluation harnesses, the analysis, and the writing.

Changes from the proposal. The central hypothesis is unchanged. Five things changed during the work, all forced by what building the system revealed. (1) *Data:* the proposal planned to train on benchmark splits, but every deterministic-grader biological benchmark turned out to be evaluation-only or withheld, so training moved to in-house controlled real-omics tasks with all benchmarks held out, yielding a clean train-XOR-eval design. (2) *Shaping:* the proposal's potential-based shaping was replaced by additive per-step-reward-to-go shaping, because a zero-terminal potential telescopes to a per-prompt constant that cancels under GRPO's group baseline. (3) *Experiment design:* the original horizon-stratified ablation was not identifiable (long tasks near-unsolvable, horizon confounded with family, scaffolding trivializing the bio decisions), so it was redesigned to a decision-focused within-family ablation with a code-capable policy, de-scaffolded tasks, and $\gamma{=}1$. (4) *Policy scale:* the policy grew from the proposal's 3–4B to a 14B data-analytic model, to obtain within-group reward variance. (5) *Scope:* the learned generative BioPRM, a proposal stretch goal, was built, trained, and run (and one of its variants is reported as a failure mode). The AI-tools policy was followed: implementation and debugging used an AI coding assistant, and the design, methodology, analysis, and writing are the author's.

References

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery, 2024. ICLR 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. PRIME.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. Also published as Nature 645:633–638, 2025.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. Infiagent-dabench: Evaluating agents on data analysis tasks, 2024. ICML 2024; DAEval, the file-based DABench eval used in this work.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, others, Aviv Regev, and Jure Leskovec. Biomni: A general-purpose biomedical ai agent. bioRxiv, 2025. snap-stanford/Biomni.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. Published at ICLR 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. Introduces Dr. GRPO.

Ludovico Mitchener, Jon M. Laurent, Benjamin Tenmann, Siddharth Narayanan, Geemi P. Wellawatte, Andrew White, Lorenzo Sani, and Samuel G. Rodriques. Bixbench: a comprehensive benchmark for llm-based agents in computational biology, 2025. FutureHouse; carries a benchmark canary. Author list partially inferred — confirm before final.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 278–287, 1999.

Zhuoyang Qiu et al. Rewarding the scientific process: Process-level reward modeling for agentic data analysis, 2026. DataPRM; code at github.com/zjunlp/DataMind.

Qu et al. Biomnibench: Process-level evaluation of llm agents for real-world biomedical research. bioRxiv, 2026. BiomniBench-DA; data HF phylobio/BiomniBench-DA; gated. Full author list PENDING.

Yusuf Roohani, Andrew Lee, Qian Huang, Jian Vora, Zachary Steinhart, Kexin Huang, Alexander Marson, Percy Liang, and Jure Leskovec. Biodiscoveryagent: An ai agent for designing genetic perturbation experiments, 2024. ICLR 2025; snap-stanford/BioDiscoveryAgent.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. Introduces GRPO.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. NeurIPS 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2023. Published at ACL 2024.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents, 2024. ICML 2024.

Kenny Workman et al. scbench: Evaluating ai agents on single-cell rna-seq analysis, 2026. 394 verifiable problems; deterministic graders; latchbio/scbench. Full author list PENDING.

Zhiheng Xi et al. Agentprm: Process reward models for llm agents via step-wise promise and progress, 2025. ID from repo proposals; web re-verification PENDING.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022. ICLR 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and others Jin. Dapo: An open-source llm reinforcement learning system at scale, 2025.

Ningyu Zhang et al. Scaling generalist data-analytic agents, 2025. DataMind; DataMind-12K data and DataMind-7B/14B models; code github.com/zjunlp/DataMind.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. Genprm: Scaling test-time compute of process reward models via generative reasoning, 2025. Published at AAAI 2026.

A Additional experiments and training details

SFT cold-start. A Qwen3-4B-Instruct policy was LoRA-SFT'd on 2,824 DataMind analysis trajectories with assistant-only loss masking (2 epochs, lr 10^{-4}), reducing training loss $1.60 \rightarrow 0.59$ in about 32 minutes on one H100. This model is the cold-start for the smoke rig and the base of the learned verifier; the reported ablation policy is the 14B model.

Single-turn loop validation. Before the multi-turn bio agent, the GRPO loop was validated on a DataMind/TableBench diagnostic: held-out accuracy rose 0.15→0.25 over 40 steps, with only 7 of 40 steps producing a gradient, an early signal that within-prompt reward variance is the binding constraint.

Verifier training data. The bio verifier SFT set is 1,288 step-examples ($\{1.0{:}1035,\ 0.5{:}35,\ 0.0{:}218\}$); mixing in upstream DataPRM execution-grounded traces yields 3,988 examples ($\{1.0{:}1935,\ 0.5{:}935,\ 0.0{:}1118\}$, bio fraction 0.32). The generic verifier is the same recipe on 3,990 upstream-only, class-balanced examples. Trajectories: 216 positive and 96 style-matched negative agent runs, plus 16 deterministically corrupted negatives.

B Implementation and tests

The system is implemented in Python with transformers/peft for the model, vLLM for rollouts, and jupyter-style `exec` for the kernel; the GRPO objective uses no RL library. Forty-seven unit tests pass (10 GRPO math; 9 reward invariants including the telescoping lemma, prefix-causal scoring, and the strict score parser; 7 multi-turn including a full real-task rollout and the reward-hacking guards; 6 outcome/grader and judge-routing safety; 5 registry contamination rules; 4 learned-verifier batching; 3 parser; 2 bio-real split disjointness; 1 loader). Every rollout logs its git SHA, resolved config, and seed for reproducibility.