

Research results

27 October 2026

[PDF report](#) · [HTML report](#)

Learning to write statistical models

In the coral task, GLM-5.3-Flash ran a program in PyMC, a Python library for probability models, and submitted a linear regression to predict the maximum diameter of brain coral at reef sites ([coral training example](#)).

Coral episode	Measurement
Predictors	Water depth; sedimentation rate; years since bleaching
Candidate rows / requested by ID	48 / 8
Prompt	Summaries of the observed outcomes from all 48 candidate sites
Withheld-row fit	A separate verifier fitted the regression
Utility / admission / predictive fit	2.2276 / 1.5 / 0.7276

An attempt at a task, which we call an episode, can include data requests and a program submission. A query budget caps how many observations the agent can request. The program is a probabilistic model of how the data arose, and it predicts the rows withheld from the agent. SynthStats is designed to train language models to write such programs, the rewards updating their choice of queries as well as their programs.

The verifier checks the syntax and safety of a program before fitting it, and checks the built model under a strict PyMC profile, which excludes custom densities, arbitrary additions to the log density, priors and branches that depend on PyMC data containers. An accepted program receives a utility of $1.5 + q$ if its held-out expected log predictive density (ELPD) is finite. ELPD measures the predictive density assigned to withheld observations; q is the hyperbolic tangent of its improvement over the task's baseline per held-out row. A refused program or an episode without a submitted program receives zero.

What the demonstrations show

Both tasks come from BoxingGym, a benchmark of simulated scientific environments, and in each task the language model received a description in English together with 80 observations of one simulated dataset, the other 20 being held out for scoring.

In both demonstrations, dugong growth and disease spread, the agent was the language model Qwen3.6-35B-A3B, which wrote eight programs for each: four single attempts and four proposals from a two-round loop in which a critic gives feedback on fitted predictions. We wrote the reference knowing how the simulator generates its data. All eight dugong-growth programs scored above the floor model (which uses the same predictive distribution whatever the input), three of them with point estimates above the reference program on this simulated dataset of 20 held-out points. The Monte Carlo standard error (MCSE) measures the error from estimating a score with 200 posterior draws. Single attempts 2 and 4 scored higher by 1.4059 (MCSE 0.1048) and 1.3761 (0.1051), each more than two MCSEs. Loop proposal 4 also scored above the reference, by 0.0907 (MCSE 0.1355), within two MCSEs. Six of the eight disease-spread programs received scores; all six beat the floor model and scored below the reference. Each figure shows the reference next to the first single attempt, that is, the program written in a single call to the model without any feedback.

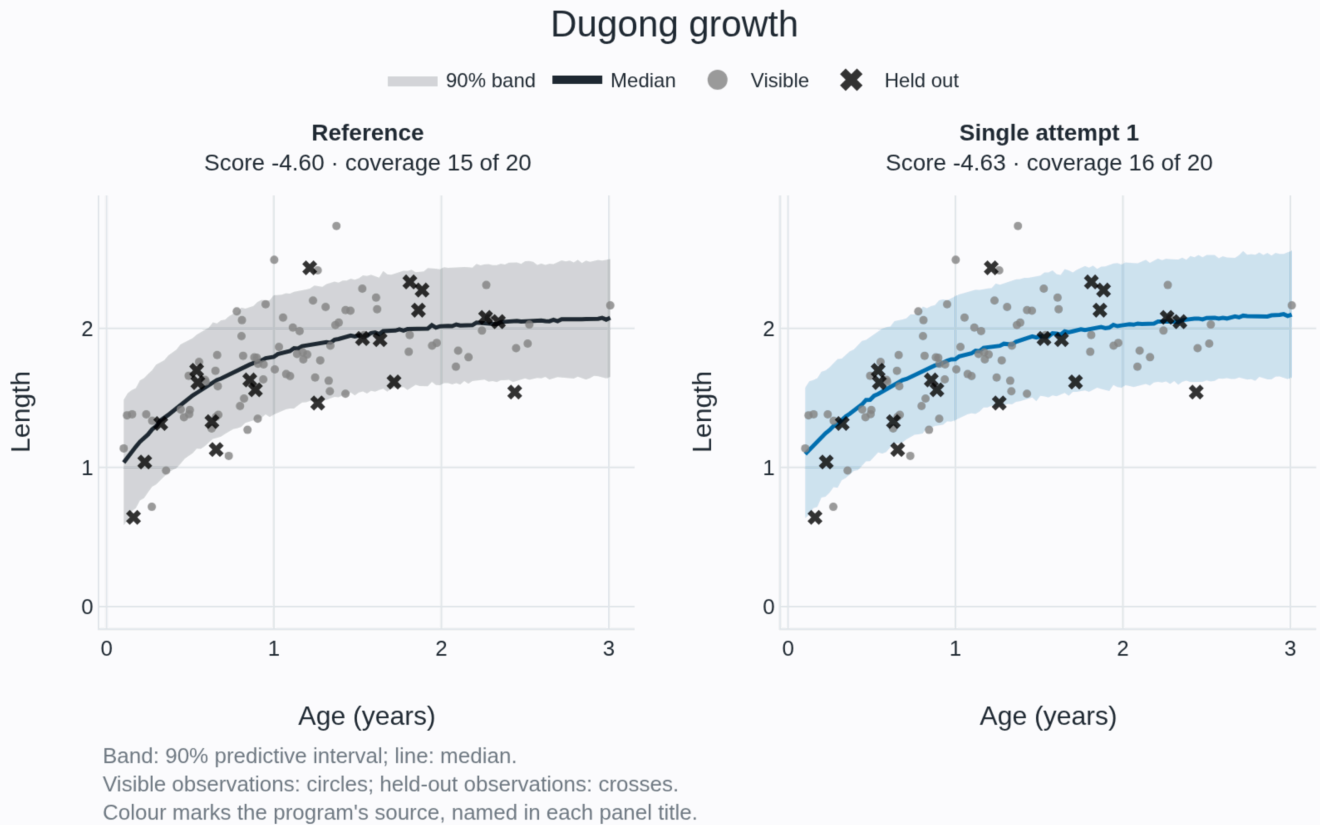


Figure 1. Dugong growth. We plot the simulated observations as circles where the agent saw them, and crosses where we held them out. Growth slows with age in the fits of both the reference and the first single-attempt program. We draw a line at the predictive median, with a band containing the central 90% of predictive draws. Panel scores are summed held-out log predictive densities (higher is better); coverage counts the held-out observations inside the 90% predictive interval.

Disease spread

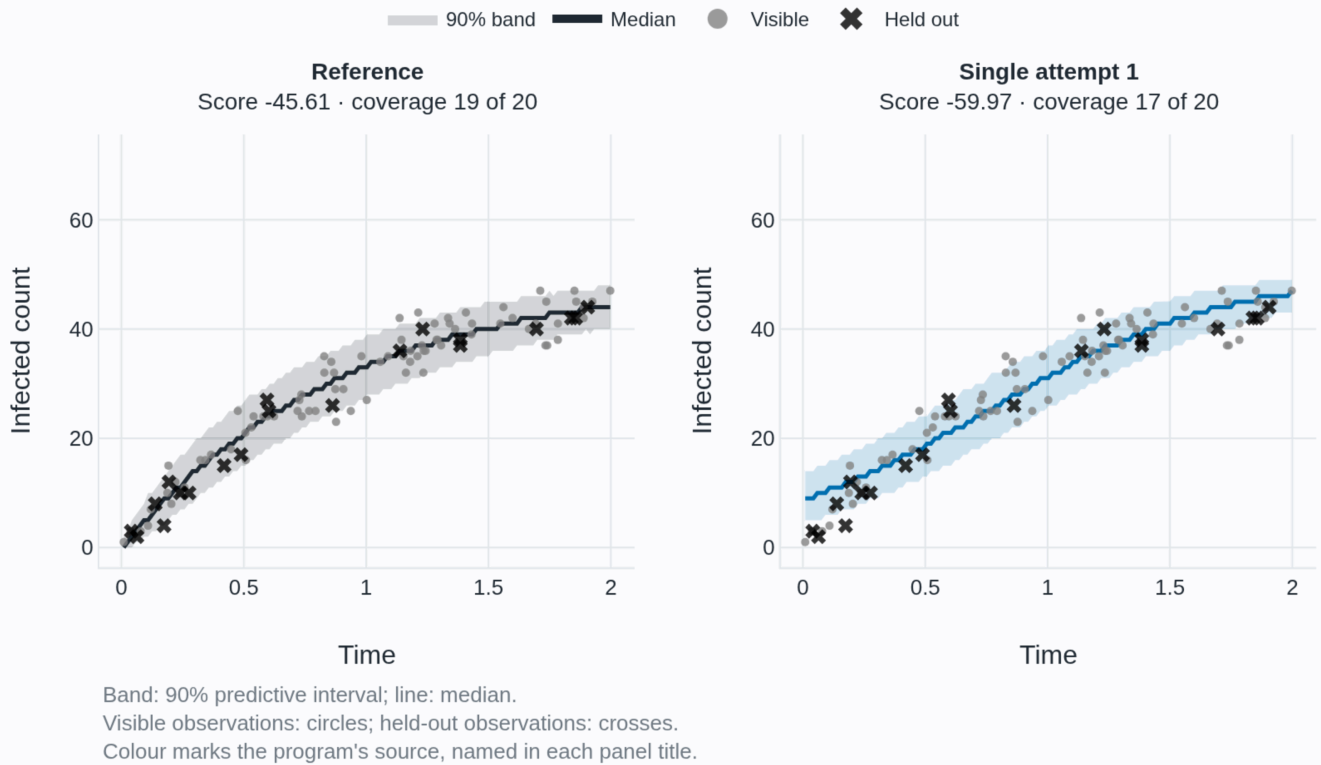


Figure 2. Disease spread. We fitted the first single-attempt program to the simulated demonstration data: the agent saw the circles, and we held out the crosses. Its fit follows the rise in infected counts. It scores below the reference. Lines give the predictive medians, and bands the central 90% intervals. Panel scores are summed held-out log predictive densities (higher is better); coverage counts the held-out observations inside the 90% predictive interval.

Demonstration fitting and generation

Consider the dugong reference at age 0.16. We fitted it in a single chain, with 200 tuning iterations followed by 200 retained posterior draws, and repeated each retained draw 20 times for prediction, which gives 4,000 simulated lengths at that age; its 90% predictive interval runs from the 5th to the 95th percentile of those lengths, 0.69 and 1.55. We fitted and sampled every other scored demonstration program the same way. We calculated MCSEs from batch means of each draw's delta-method influence on the total held-out log score, over ten consecutive batches of 20 draws; MCSEs of differences combine the two fits in quadrature, treating them as independent.

The programs that the language model wrote came from calls of at most 2,048 output tokens each, in the critic loop as in the single attempts (the demonstration records give the model calls, tokens and timings for each activity).

Principal findings

We tested four constructed forgeries against one verifier version: all four were refused at model build, and an honest control scored. Our UAI 2026 paper on likelihood hacking, Likelihood hacking in probabilistic program synthesis, appears in the UAI 2026 proceedings.

The training comparisons

We trained the larger model, GLM-5.3-Flash, on 220 packages, with four episodes on each, as in the coral task above. For each task, the package contains its prompt and data, the withheld rows and reference scores. Two checkpoints of that run are compared with the untrained model on 103 evaluation packages from 39 scenario groups held out from training: the checkpoint after three updates, and the final checkpoint after fourteen updates. On each package, we subtract the untrained model's mean utility from the trained model's, which gives the paired difference. Each 95% interval comes from 10,000 resamples of the 39 whole groups.

Training run	Setting
Training objective	the Tinker cookbook baseline: each episode's reward is compared with the rewards of the other episodes on the same task, with no penalty for moving away from the untrained model
Training objective of our <u>16 October pilot</u>	reference-weighted trajectory balance
Updates	14
Episodes	880
Low-rank adapters	rank 32, at a learning rate of 1e-5

Checkpoint after three updates

Checkpoint 000003 against the untrained model	Setting
Updates before the checkpoint	3
Episodes per package, for each model	2

Final checkpoint after fourteen updates

Final checkpoint against the untrained model	Setting
Updates before the checkpoint	all 14
Episodes per package, for each model	8

Judge-reward comparison

Judge-reward arm against its matched untrained model	Setting
Training	a separate arm, on the same packages and updates as above
Training reward	fit reward of weight 1.0, plus a language-model judge term of weight 0.25
Evaluation reward	held-out fit, with the judge disabled, using this experiment's own scorer
Evaluation packages	all 103, from 39 groups
Episodes per package, for each model	8

The fit-only and judge-assisted comparisons sampled their untrained episodes separately and used different verifier versions.

Small-agent training

The smaller model is Qwen3.5-0.8B, used as a tool-using agent. We first gave it a supervised warm start, which trains it to imitate positively rewarded episodes from teacher language models. From that one warm start, we trained it with Subtrajectory Balance (SubTB) in two arms, at learning rates of 1e-5 and 1e-6 (one tenth of the first rate), and with group relative policy optimization (GRPO) in a third, also at 1e-5, each over eight updates on 220 training packages in 81 scenario groups. SubTB is computed on segments of the agent's sequence of actions, and GRPO compares an episode's reward with those of the other episodes on the same task. Every arm is measured against the warm start it began from.

Small-agent training	Setting
Training stack	Harbor, mini-SWE-agent and SkyRL
Supervised warm start	193 verified teacher episodes; 1,930 tool actions; 63 of the 81 training-side groups; two epochs; 98 updates
Reward training	Eight updates in each of three arms: SubTB, GRPO, and SubTB at a learning rate of 1e-6
Low-rank adapters	Rank 32

Small-agent comparison with the warm start	Design
Held-out evaluation packages	103, in 39 scenario groups
Arm	The policy after eight updates, compared on each package with the warm start it began from
Paired change	Mean utility after eight updates minus the warm start's mean utility
Interval	95%, from 10,000 resamples of whole scenario groups
Three comparisons	Bonferroni interval at 98.333%, for three comparisons sharing one warm start

Full supporting results.