MEASURING INTELLIGENCE BEYOND HUMAN SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

How can we measure intelligence beyond human capability?
Human-authored benchmarks saturate, and above human capability, examiners may not know which tasks are both hard and verifiable. We argue that this difficulty is inherent to absolute-scale evaluation and propose a new paradigm based on relative measurement in which models generate public challenges that separate other systems. We develop a novel protocol that naturally scales with agent capabilities, requires no human-authored questions, and is also judge-free. We instantiate this framework across verifiable and open-ended, non-verifiable domains, illustrating how model-generated evaluation can continue to measure systems beyond the human frontier.

1 INTRODUCTION

How should intelligence be measured? The question is older than the study of artificial intelligence.

The modern scientific study of intelligence began with the work of Charles Spearman (Spearman, 1904). Spearman observed that individuals who performed well on one cognitive test tended also to perform well on many others, a phenomenon now known as the positive manifold. He hypothesized that these positive correlations were explained by a single latent factor, which he called **general intelligence (g)**. This marked a fundamental shift from viewing intelligence as a collection of observable abilities to modeling it as a latent statistical variable inferred from observable behavior, laying the foundations of modern psychometrics. Interestingly, this is also the seed of factor analysis in statistics.

Most AI benchmarks inherit this psychometric structure. Systems are evaluated on tasks supplied by humans or environments fixed in advance: game environments, broad academic benchmarks and so on (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023; Chollet, 2019; White et al., 2025; Glazer et al., 2024; Phan et al., 2026; Kiela et al., 2021). This becomes a scaling problem at the frontier: benchmarks saturate, and new ones require humans to generate tasks that are difficult and verifiable. Above human expert level, the bottleneck is not only the number of questions, but the examiner's ability to formulate discriminating questions.

Relative intelligence measurement. A different starting point is relative measurement. Turing's imitation game measures machine intelligence by comparison with human behavior before a human judge, rather than by an absolute score on a fixed test (Turing, 1950). Modern pairwise preference and arena methods similarly compare systems through judged interactions (Zheng et al., 2023; Chiang et al., 2024). These protocols point toward relative evaluation, but the tasks or judgments are still largely human-supplied. A notable exception is MathDuels (Xu et al., 2026), which explores model-generated mathematical challenges solved across the model population, as an evaluation primitive.

However, purely pairwise comparative protocols are vulnerable to failure modes that can make the resulting signal uninformative. A model may exploit private information, construct trapdoor tasks, or target one particular weakness of a single opponent rather than reveal a general capability distinction, see more details in section 3.1.

A different approach to measuring intelligence. We propose a model-generated and separative approach to intelligence evaluation, which we call *adversarial psychometrics*. While classical psychometrics measures intelligence by how well subjects solve examiner-designed tasks, adversarial psychometrics evaluates a model's ability to discover capability boundaries among other systems.

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.

We hypothesize that a model displays an important component of intelligence by constructing challenges that separate a population of solvers.

Our central move is to replace pairwise comparison with separative measurement. A challenge is evaluated not by whether it defeats one opponent, but by the variance in confidence it induces across a population of solvers. This shift rewards proposers for identifying distinctions across systems, rather than for exploiting specific weaknesses of a target. By evaluating systems on their capacity to discover such capability gaps, we obtain a richer statistical signal that can scale beyond the human frontier and can support sparse, or even judge-free adjudication.

Paper overview. The paper proceeds in three stages. First, we show that natural pairwise challenge protocols are fundamentally susceptible to private-information and trapdoor attacks, motivating a new one-to-many formulation. Second, we develop the separative psychometric framework, including its scoring rule, adaptive weighting, and judge-free adjudication mechanism. Third, we demonstrate the framework empirically across verifiable and open-ended, non-verifiable domains, showing that model-generated challenges discriminate between current systems without a semantic judge, correlating well with existing intelligence indices.

1.1 Relation to existing work

Psychometric and adaptive-testing models infer latent ability from responses to diagnostic items designed and administered by human examiners (Spearman, 1904; Rasch, 1960; Lord, 1980; van der Linden & Glas, 2010). Paired-comparison systems such as Bradley–Terry and Elo infer relative ratings from sparse pairwise outcomes. Methods such as TrueSkill extends probabilistic skill inference to multiplayer and team results and represents uncertainty. (Bradley & Terry, 1952; Elo, 1978; Herbrich et al., 2006). Modern AI evaluation has focused on benchmark construction, contamination resistance, frontier-level difficulty, dynamic task collection, and pairwise comparison, including MMLU, BIG-Bench, HELM, ARC, LiveBench, FrontierMath, Humanity's Last Exam, Dynabench, MT-Bench, and Chatbot Arena (Hendrycks et al., 2021; Srivastava et al., 2023; Liang et al., 2023; Chollet, 2019; White et al., 2025; Glazer et al., 2024; Phan et al., 2026; Kiela et al., 2021; Zheng et al., 2023; Chiang et al., 2024).

Language-Model-as-an-Examiner, Auto-Arena, and PeerRank automate question generation and evaluation while retaining semantic judgments as the source of their scores (Bai et al., 2023; Zhao et al., 2025; Margalit et al., 2026). LivingArena targets opponents' weaknesses using a separate three-model judging panel (Chen et al., 2026). MathDuels (Xu et al., 2026) is particularly related to our work. It proposes a mathematical self-play benchmark in which models act as both problem posers and solvers. Our work shares the idea that task generation itself is a measurable capability, but differs in several respects: we study arbitrary domains rather than mathematics, develop a psychometric framework based on population separation against challenger-committed answers and investigate protocols that don't require a semantic judge.

Relation to Scalable Oversight. The scalable oversight problem asks: how do we obtain reliable supervision for systems whose competence exceeds that of their supervisors (Amodei et al., 2016; Christiano et al., 2018). Existing proposed mechanisms include debate, in which models argue opposing answers before a weaker judge (Irving et al., 2018), weak-to-strong generalization, in which a weak supervisor elicits a stronger model's latent ability (Burns et al., 2024), and sandwiching-style experiments, in which non-experts use model assistance on tasks evaluated against expert-defined answers (Bowman et al., 2022). Our work shares the same motivation: human evaluators may eventually be unable to author or verify sufficiently discriminating tasks. However, whereas scalable oversight seeks reliable supervision of individual systems, we focus on comparative measurement.

Relation to interactive proofs and complexity theory. There is also a conceptual connection to multi-prover interactive proofs (MIPs). In an MIP, a bounded verifier extracts reliable evidence by comparing the behavior of multiple provers; the theorem of Babai et al. (1991) that $\mathsf{MIP} = \mathsf{NEXP}$ demonstrated the remarkable power of such multi-agent verification. Our protocol is not an MIP in the formal sense: we do not provide completeness or soundness guarantees for a fixed language, nor do we assume a polynomial-time verifier. However, we have the shared idea that multiple agents can reveal information that a single agent cannot. Whereas MIPs use cross-prover consistency to verify mathematical claims, adversarial psychometrics uses cross-model disagreement to measure capability boundaries.

2 BACKGROUND: SPEARMAN'S GENERAL INTELLIGENCE g

In the early 1900s, experimental psychologists were intrigued with the question of measuring human intelligence. Spearman followed Galton's theories of intelligence but noted that Galton surprisingly failed to find associations between different performance metrics and common indicators of intelligence.

Spearman's experiment. Spearman set out "to determine this Intelligence in a definite objective manner, and to discover means of precisely measuring it" (Spearman, 1904, p. 206). His clearest data came from a preparatory school for boys. He ranked the pupils by their results in the regular school examinations in Classics, French, English and Mathematics. He also tested their discrimination of pitch and obtained a ranking of musical ability for the 22 boys taking music lessons (Spearman, 1904, pp. 248–250, 270). Figure 1(a) is his table of correlations among these six measures for those 22 boys (Spearman, 1904, pp. 275, 291).

The positive manifold. Spearman observed that every correlation in the table is positive. A boy who ranks high in one subject tends to rank high in all of them, and even to hear finer differences in pitch. The entries are also orderly, falling steadily "as the eye travels either to the right or downwards," a pattern Spearman called the *hierarchy* (Spearman, 1904, p. 275). He concluded that "all branches of intellectual activity have in common one fundamental function," plus a specific element unique to each (Spearman, 1904, p. 284). He called it General Intelligence, later written g.

	Classics.	French.	English.	Mathem.	Discrim.	Music.
Classics,	*0.87*	0.83	0.78	0.70	0.66	0.63
French,	0.83	*0.84*	0.67	0.67	0.65	0.57
English,	0.78	0.67	*0.89*	0.64	0.54	0.51
Mathem.,	0.70	0.67	0.64	*0.88*	0.45	0.51
Discrim.,	0.66	0.65	0.54	0.45		0.40
Music,	0.63	0.57	0.51	0.51	0.40	

(a) correlations between tests (p. 275)

Activity.	Correlation with Gen. Intell.	Ratio of the common factor to the specific factor.
Classics,	0.99	99 to 1
Common Sense,	0.98	96 4
Pitch Dis.,	0.94	89 11
French,	0.92	84 16
Cleverness,[3]	0.90	81 19
English,	0.90	81 19
Mathematics,[4]	0.86	74 26
Pitch Dis. among the uncultured,[5]	0.72	52 48
Music,	0.70	49 51
Light Dis.,[6]	0.57	32 68
Weight Dis.,[6]	0.44	19 81

(b) correlation with g (p. 276)

Figure 1: Spearman's 1904 tables (Spearman, 1904). (a) The 22 music pupils of a preparatory school with the diagonals giving reliabilities (how well two separate examinations in that subject agree). (b) Estimates of correlations with g pooled across his experiments.

The one-factor model. In mathematical terms, person i's score on test j is $s_{ij} = \mu_j + \lambda_j g_i + \varepsilon_{ij}$: one ability g_i per person, one loading λ_j per test, and a test-specific remainder, so the person-by-test score matrix is approximately rank one, $X \approx g\lambda^\top$. If the specific parts are uncorrelated, the correlation between two tests is a product, $r_{jk} = \ell_j \ell_k$ for $j \neq k$, where $\ell_j = \mathrm{corr}(\text{test } j, g)$. The correlation matrix is then rank one off its diagonal, and sorting tests by ℓ_j yields the hierarchy.

Correlation with general intelligence. Figure 2 shows that Spearman's table is approximately rank one: it is the product of a single vector with its own transpose, $\ell\ell^\top$ with $\ell = (0.96, 0.88, 0.80, 0.75, 0.67, 0.65)$, plus a small residual. Each ℓ_j is test j's correlation with g. Classics and French correlate most strongly with g (0.83) while pitch and music correlate least (0.40).

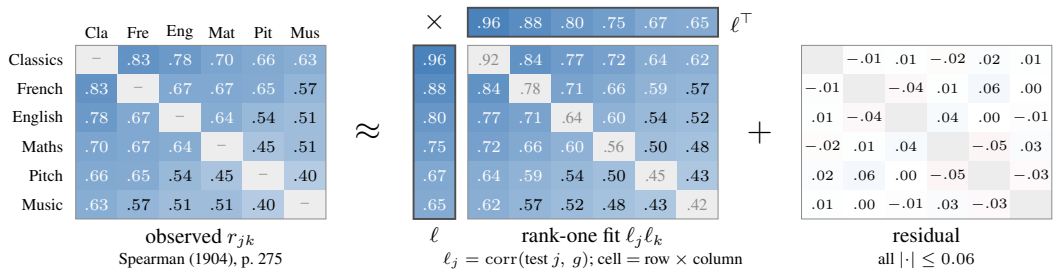

Figure 2: Spearman's correlations $\approx$ rank-one $\ell\ell^\top$ + small residual; $\ell_j = \mathrm{corr}(\text{test } j, g)$.

Section 5.2 estimates a benchmark g using this approach and reports its correlations with SepaRank.

3 WARM-UP: PAIRWISE INTELLIGENCE TOURNAMENTS

A natural first attempt is a pairwise intelligence tournament. Given two systems i and j, system i is asked to produce a challenge that it can solve but system j cannot. The challenge must be publicly specified and objectively resolvable. System j then submits an answer under the same resource budget, and the outcome is recorded as a win for i if i's answer is correct and j's answer is incorrect.

Over many such matches, the outcomes define a directed comparison graph: an edge $i \to j$ indicates that system i found a challenge separating itself from system j. In principle, this graph could be aggregated by any standard paired-comparison model, such as Bradley–Terry, Elo, or TrueSkill. For example, a Bradley–Terry model assigns each system a latent score θ_i and models

$$\Pr(i \text{ beats } j) = \sigma(\theta_i - \theta_j).$$

Equivalently, one could maintain Elo-style online ratings from the same pairwise outcomes.

This gives a simple self-scaling measurement scheme: stronger systems should be able to find challenges that separate themselves from weaker systems, and the rating scale evolves with the population being measured. However, the pairwise formulation has fundamental flaws.

3.1 CAVEATS OF PAIRWISE PROTOCOLS

The first difficulty is that this naive protocol rewards separation from a single opponent, and therefore creates incentives to exploit asymmetric information rather than capability. The simplest attack is a private-state question, such as "which bit am I thinking of?"

One might try to rule out such private or otherwise unfair questions. But the same problem can arise even in questions that appear fair. For example, a challenger can generate two large primes, publish their product, and ask for its factorization. Under the same resource budget the task may be hopeless for the solver, while the challenger has "solved" it only because it inserted the trapdoor during construction.

The common issue is that the challenger possesses information unavailable to the solver: in one case a private state, and in the other an algorithmic backdoor. Such separations need not reflect superior intelligence.

A second problem is adjudication. Even if private-information and trapdoor questions are formally disallowed, the protocol still requires someone to decide whether a challenge is admissible and whether its proposed resolution is correct. Without an external judge, the parties' own reports can enforce agreement but not truth. Penalties for disagreement may induce the players to agree, but they cannot distinguish the truthful resolution from a biased one. Thus pairwise protocols either rely on external adjudication, which limits scalability, or remain manipulable.

The key observation is that private information and trapdoors are powerful mainly because the challenger is trying to fool a single opponent. If the same challenge is posed simultaneously to many independently trained systems, the reward depends on population disagreement rather than success against one solver. In an appropriately designed mechanism, shared uncertainty among players is indicative of an uninformative challenge and should yields no separation reward. This motivates the one-to-many protocol developed next.

4 THE BASIC CHALLENGE PROTOCOL

We call our novel ranking protocol **SepaRank** , and its basic description is as follows. A proposer submits a binary question with answers *A* and [1]. Each solver returns an answer and a confidence, equivalently a posterior probability p_i assigned to answer *A*. The proposer is rewarded for inducing high variance in the values $\{p_i\}$ across solvers. No restriction is placed on the semantic content of the question. The question may be factual, mathematical, philosophical, subjective, future-contingent, or otherwise difficult to resolve externally.

[1] Binary questions are without loss of generality. Any finite-answer question can be reduced to a sequence of binary questions (e.g., by asking for successive bits of the answer or by using a binary search over the answer space). Since our objective is to measure discriminative power rather than answer format, we restrict attention to binary questions for simplicity.

The variance scoring changes the incentive relative to pairwise challenge protocols. The question "which bit am I thinking of?" no longer succeeds: a calibrated solver has no evidence and should report $p_i = \frac{1}{2}$, so the variance is small. Likewise, a trapdoor-ed factoring instance does not help merely because the proposer generated the primes; the proposer anyways is not required to solve the problem, and if the problem is too hard, calibrated reports again concentrate near $\frac{1}{2}$. The proposer succeeds only by finding questions that genuinely separate the audience. We conjecture that this discourages some uninformative challenges.

We use variance as the proposer objective for simplicity. Other separation objectives are also possible, such as the empirical entropy of the distribution of reports or a weighted Jensen–Shannon divergence, which measures how much information a report carries about which solver produced it.

4.1 Protocol

We evaluate systems in two roles: proposer and solver. The Separank protocol constists of R rounds. For an N-model panel, each round consists of N matches. In match i, model i proposes one binary question and privately commits to an answer $y \in \{0, 1\}$. Every other model answers that question. All N matches during round r use the full information of the game of up to round $r - 1$.[2] Current cross-match commitments and solver answers remain hidden until the round is complete. Only then are scores, weights, and public history updated. That way, matches can run in parallel.

Models use stable pseudonyms, which identify their answers and commitments throughout the entire game. Each solver i reports a probability $p_i \in [0, 1]$ that the committed answer is 1. Thus $p_i = 1/2$ expresses equal uncertainty between the two answers, while values near 0 or 1 express confidence.

The proposer is rewarded for separating the solvers' beliefs. For one question, let $p_1, \ldots, p_m$ be the usable reports and let $w_1, \ldots, w_m$ be their weights, normalized to sum to one. These are the round-start solver weights, renormalized over the solvers whose reports are usable. The proposer reward is

$$\mu = \sum_{i=1}^{m} w_i p_i, \qquad P = 4 \sum_{i=1}^{m} w_i (p_i - \mu)^2.$$

This is the weighted population variance of the reported probabilities. With equal weights, it reduces to ordinary population variance. The factor 4 places the reward in $[0, 1]$: it is zero when all solvers report the same probability, and reaches one when half the weight reports 0 and half reports 1.

Variance captures the distinction between a question that separates the population and one that leaves everyone equally uncertain. For example, suppose a proposer asks, "which fair coin flip did I privately observe?" If no solver has information about the flip, each should report $1/2$, giving the proposer zero reward. The proposer's private knowledge alone therefore produces no separation in this case. A public question may instead give some solvers a basis for confidence while leaving others uncertain, producing positive variance. The objective rewards this difference in reported beliefs, rather than difficulty alone.

Solvers are scored separately against the author's committed answer:

$$L_i = (p_i - y)^2.$$

This proper scoring rule is Brier loss, and is smallest for an accurate, confident report and largest for a confidently incorrect one. For a fixed commitment, a solver minimizes its expected one-step loss by reporting its probability that $y = 1$ (Brier, 1950; Gneiting & Raftery, 2007). The scoring target throughout the experiments is the commitment: even in verifiable domains, such as Python code challenges where execution could produce the correct answer, the solvers are scored against the proposer-committed bit. The two roles therefore ask different things of a model: find questions that separate the population, and forecast other authors' committed answers.

Scores accumulate as two separate averages. A model's proposer score, $\bar{P}$, is the mean of its scored proposal rewards; its solver loss, B, is the mean of its scored response losses. Higher $\bar{P}$ and lower B are better. For the final ranking, we combine the two components with equal weight:

$$J = \frac{\bar{P} + (1 - B)}{2}.$$

[2] In practice, we truncate the protocol transcript to the past 5 rounds for computational feasibility. However, model scores and standings are computed from round 1 through $r - 1$.

This joint score is computed after play, separately for code and open-ended games. We also report both components. During play, adaptive weights depend only on past solver losses, as described in Section 4.2.

4.2 Adaptive solver weights

With equal weights, a proposer can continue to earn a high reward by separating the same groups of models. Once a question distinguishes the stronger solvers from the rest, variations of that question may keep producing disagreement without revealing differences among the stronger solvers themselves. We introduce adaptive weights to shift more of the measurement toward models that have performed well as solvers, and push proposers to keep separating the frontier. The weights determine how much each solver's report contributes to the proposer's variance reward. Every eligible model still solves every question from another author. A model with a small weight therefore retains the same opportunities to improve its solver score.

We begin with 3 warmup rounds in which all N active models have equal weight, $W_i = 1/N$. This gives the game an initial period of observation before past performance affects the proposer reward. After warmup, we use each model's cumulative mean solver loss B_i through the last completed round to set its weight for the next round:

$$W_i = 0.95 \frac{\exp(-B_i/0.25)}{\sum_{j=1}^{N} \exp(-B_j/0.25)} + \frac{0.05}{N}.$$

Lower solver loss gives greater weight. The denominator 0.25 controls how sharply the weights respond to differences in loss: a smaller value would concentrate more weight on the best-performing solvers. The remaining 5% is distributed uniformly, ensuring that every active model retains positive influence. Weights are recomputed from these cumulative losses after each completed round, For each question, we exclude its author and renormalize the weights over the usable solver reports, as in the variance formula of the protocol.

The intended effect is to make broad, already-established separations less rewarding as weight shifts toward the better forecasters. If most of the weight lies on solvers that give similar reports, disagreement confined to low-weight solvers contributes little to the reward. A question that separates the higher-weight solvers can contribute more. We conjecture that this encourages finer distinctions among reliable forecasters. The rule does not explicitly reward novelty, as this would require a separate definition of "novelty" and a judge to enforce it.

4.3 Resolution rules and optional adjudication

The **SepaRank** protocol separates the scoring mechanism from the resolution rule. In every round, solvers submit probabilities p_i, the proposer is scored by the induced separation among those probabilities, and solvers are scored by a proper scoring rule against a resolved bit y. The resolution rule specifies how y is obtained.

In mechanically verifiable domains, y may be determined by executing a program, checking a proof, or verifying a certificate. In this case solver loss measures accuracy with respect to an objective public truth. In non-verifiable domains, y may instead be the proposer's committed answer, along with a supplied rationale checked by a judge. In SepaRank, the proposer is allowed to commit any y they wish, and the solver task becomes predicting the proposer's committed resolution. The incentives this creates for the proposer, and the extent to which models exploit them, are measured in Section D.2. In experiments where we restrict the question scope to Python programs, we can verify the true y by execution, and use it to understand the commitment behavior of proposers.

Although false commitments may increase disagreement in the short run, we conjecture that they should eventually erode trust. As a proposer becomes less reliable, calibrated solvers ought to hedge toward $\frac{1}{2}$, reducing the proposer's variance reward and creating a long-run incentive for honest commitments. The theoretical foundations of reputation, strategic stability, and conditions favoring semantically faithful commitments will be explored in future work.

Thus the proposer score always has the same meaning: it measures the ability to find questions on which the solver population separates. The solver score, however, is interpreted relative to the chosen resolution rule.

5 EXPERIMENTS

We run SepaRank in two domains. In *code games*, proposers submit a Python program with a no-argument `main()` returning 0 or 1. In *open-ended games*, they submit a yes/no question. In both, solvers predict the proposer's private commitment. Executing code checks admissibility, but its output does not replace the proposer's commitment y, used to score the solvers. Open-ended questions receive format and length checks without a content judge.

We run 10 independent games per domain, each lasting 20 rounds and containing the same ten models from five providers (Table 1). All models use their provider's low-effort reasoning setting. This was chosen to reduce API expenses and the runtime of the experiments. Every model proposes once per round, and all 9 peers solve each valid challenge. Players have stable pseudonyms within a game, and the current author's pseudonym is disclosed during a match. Prompts provide to the players the full protocol details (scoring rules, constraints, cumulative standings, etc.) and the history of the last 5 completed rounds, which includes proposed questions with their authors and commitments, and players' answers. The history is truncated to 5 rounds to reduce API expenses. Appendix B gives the full setup, and Appendix C reproduces the prompts.

Within each game, let P be the mean proposer reward and B the mean solver Brier loss. We report solver skill $S = 1 - B$ and joint score $J = (P + S)/2$, so higher P, S, and J are better. Unless stated otherwise, intervals are pointwise 95% percentile intervals from $50{,}000$ whole-game bootstrap resamples, retaining all players and both roles together.

5.1 RANKINGS AND THE TWO ROLES

Figure 3 gives separate joint-score rankings for the two domains and Table 1 reports their components. GPT-6 Astra leads both rankings, with $J = 0.762$ for code and 0.732 for open-ended games. Claude Fable 5.1 is second, with 0.640 and 0.667. Their within-game differences are 0.122 [0.085, 0.162] and 0.065 [0.045, 0.087], respectively. The middle ordering changes: GPT-5.6 Terra is third in code and fourth in open-ended games, while Gemini 3.8 Flash occupies the opposite positions.

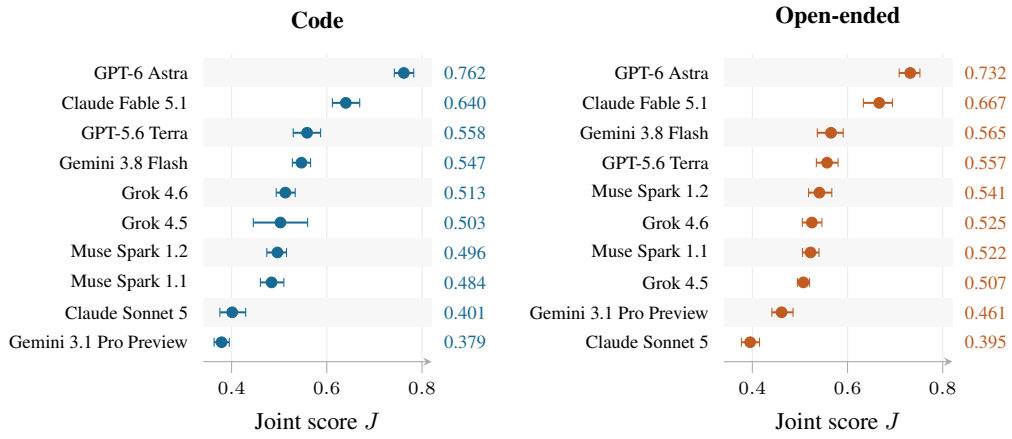

Figure 3: **Separate joint-score rankings.** Models are sorted by mean J over ten twenty-round games per domain. Bars show 95% game-bootstrap intervals.

The role profiles (Figure 4) show what the joint ranking combines. Astra has the highest proposer reward in both domains (0.629 and 0.568) and the lowest code-game Brier loss (0.105). Fable has the lowest open-ended Brier loss (0.091, against Astra's 0.104). Terra generates stronger separation than Flash in both domains, while Flash forecasts commitments more accurately. Challenge generation and solving therefore contribute distinct information about a model's performance.

The cross-domain plot (Figure 5) shows broad agreement, with Spearman $\rho = 0.915$ between the ten models' joint scores. Component correlations are 0.879 for proposer reward and 0.903 for solver skill. This agreement coexists with domain-specific differences. For example, Muse Spark 1.2 ranks seventh in code and fifth in open-ended games.

Table 1: **Roster and component means at twenty rounds.** Rows follow code rank; the last column gives open-ended rank. Figures 3 and 4 show intervals, using $S = 1 - B$ for solver skill.

			Code			Open-ended			
Rank	Model	Provider	$P \uparrow$	$B \downarrow$	$J \uparrow$	$P \uparrow$	$B \downarrow$	$J \uparrow$	Rank
1	GPT-6 Astra	OpenAI	0.629	0.105	0.762	0.568	0.104	0.732	1
2	Claude Fable 5.1	Anthropic	0.401	0.121	0.640	0.424	0.091	0.667	2
3	GPT-5.6 Terra	OpenAI	0.364	0.247	0.558	0.316	0.202	0.557	4
4	Gemini 3.8 Flash	Google	0.274	0.181	0.547	0.262	0.131	0.565	3
5	Grok 4.6	xAI	0.267	0.241	0.513	0.210	0.160	0.525	6
6	Grok 4.5	xAI	0.254	0.249	0.503	0.183	0.169	0.507	8
7	Muse Spark 1.2	Meta	0.236	0.244	0.496	0.244	0.162	0.541	5
8	Muse Spark 1.1	Meta	0.238	0.270	0.484	0.241	0.197	0.522	7
9	Claude Sonnet 5	Anthropic	0.142	0.339	0.401	0.054	0.263	0.395	10
10	Gemini 3.1 Pro Preview	Google	0.057	0.299	0.379	0.096	0.173	0.461	9

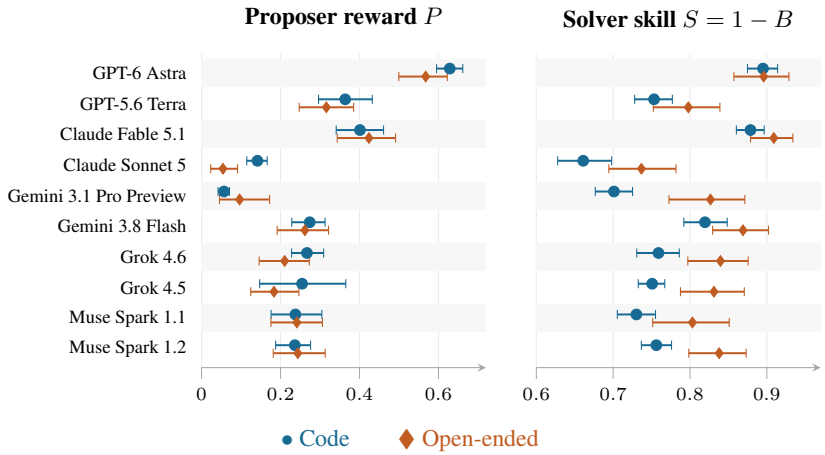

Figure 4: **The two roles.** Each row compares the same model across domains. Higher is better in both panels; bars show 95% game-bootstrap intervals.

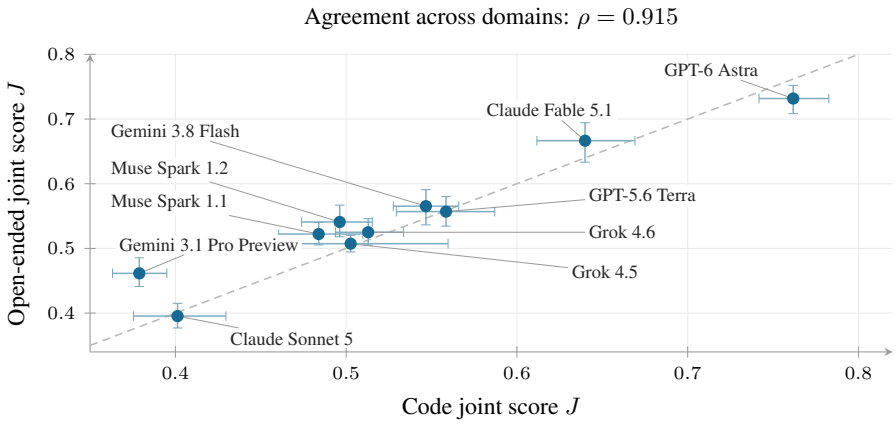

Figure 5: **Agreement across domains.** Each point is one model, with 95% game-bootstrap intervals on both axes. The dashed line marks equal scores; the rank correlation is $\rho = 0.915$.

5.2 Comparison with external benchmarks

We compare SepaRank with ten individual benchmarks, the Artificial Analysis Intelligence Index, and an estimated benchmark factor $\hat{g}$ for the same ten model releases (Artificial Analysis, 2026). Figure 6 reports Spearman correlations separately for proposer reward, solver skill, and joint score. The Index correlations are 0.900, 0.900, and 0.936 for code, and 0.790, 0.717, and 0.851 for open-ended games.

Agreement varies across benchmarks: the code joint score correlates 0.915 with Terminal-Bench 4.0, but −0.006 with SciCode. We use all ten models and assign midranks to ties.

We also estimate a benchmark g using the one-factor model in Section 2. We standardize each of the ten component benchmarks across the ten models, fit their off-diagonal correlations by minimum-residual factor analysis, and obtain a regression factor score $\hat{g}$ for each model. Correlations with $\hat{g}$ are 0.903, 0.758, and 0.915 for code, and 0.721, 0.527, and 0.758 for open-ended games, in proposer, solver, and joint order.

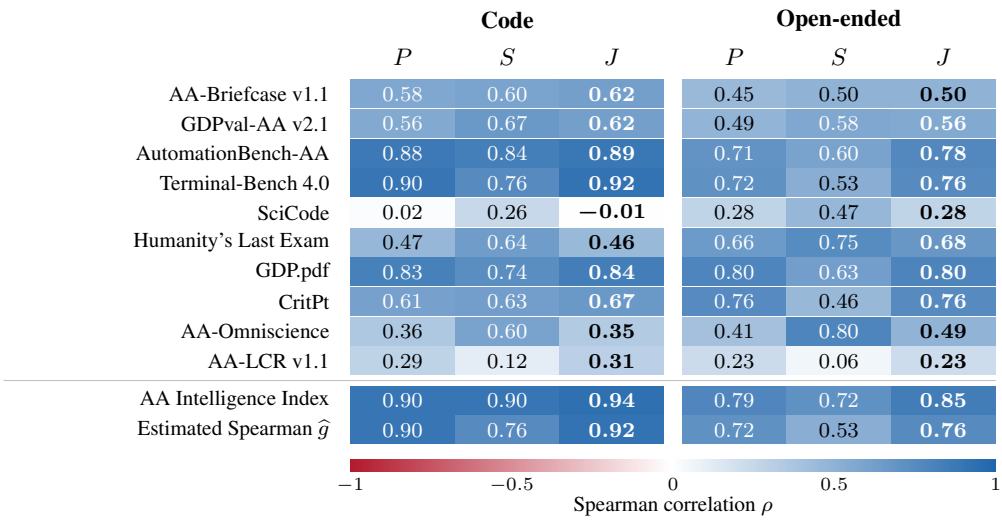

Figure 6: Spearman correlations across the ten models for proposer reward P, solver skill $S = 1-B$, and joint score J. The bottom two rows are aggregate scores: the Artificial Analysis Intelligence Index and the benchmark factor $\hat{g}$ inferred from the ten individual benchmarks. Data: Artificial Analysis v4.3.2, 23 September 2026.

We observe that our proposed SepaRank scores correlate positively with virtually all the individual benchmark measures of model capability. In addition, SepaRank scores correlate much higher with compositive measures of model capability (AA Intelligence Index), as well as the estimated spearman $\hat{g}$ from these benchmarks. Our proposed mechanism thus successfully attains our desired goal.

In Appendix D we explore the score dynamics of games, the false commitment behavior of proposers in the verifiable code setting, and the effect on attained scores of obscuring the proposer's name in a match.

6 Conclusion

We proposed an adversarial tournament protocol for measuring advanced AI capabilities. Models are evaluated by their ability to generate challenges that separate the forecasts of peer solvers. The resulting separations and ability to solve the challenges induce a ranking of the agents that we call **SepaRank** . The underlying thesis is that when human examiners can no longer reliably design frontier-level tests, evaluation itself must become model-generated and incentive compatible.

As for further work, the protocol described induces machine-generated questions and knowledge that can be further used for post-training and self-improvement. An important direction is to ensure diverse and effective generation of questions as well as correct answers at the frontier of knowledge.

AI USE STATEMENT

The LLMs evaluated in this study generated challenges and answers as participants in the experimental protocol. Their outputs constitute the experimental data. Separately, we used OpenAI Codex to assist with implementing and testing the experimental software, processing experimental records and external benchmark data, statistical analysis, preparing figures and tables, and formatting the paper in LaTeX. The authors take responsibility for the final text, methods, results, and claims.

ETHICS STATEMENT

This study evaluates LLMs and did not recruit human participants. Its goal is to improve the measurement of model capabilities through model-generated challenges. Such challenges may contain incorrect or biased content. Their inclusion as experimental observations does not endorse their claims. The resulting rankings describe performance within the evaluated protocol and should not be treated as certification of a model's safety, trustworthiness, or suitability for high-stakes deployment.

REPRODUCIBILITY STATEMENT

Section 5 describes the experimental design and reported measurements. Appendix B documents the model identifiers, generation settings, random seeds, scoring rules, information available to players, and handling of invalid responses. Appendix C provides the prompts used in the experiments. We retain the experimental responses, game states, and analysis inputs to support verification and recomputation of the reported results. Exact regeneration of model responses may vary with stochastic sampling and changes to provider-hosted models.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*, 2016. URL https://arxiv.org/abs/1606.06565.

Artificial Analysis. Artificial analysis intelligence benchmarking methodology, 2026. URL https://artificialanalysis.ai/methodology/intelligence-benchmarking. Version 4.3.2. Benchmark values and model comparison pages accessed 23 September 2026; source URLs, published settings, and extracted numerical values archived in the accompanying analysis.

László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, Mar 1991. ISSN 1420-8954. doi: 10.1007/BF01200056.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, Jiayin Zhang, Juanzi Li, and Lei Hou. Benchmarking Foundation Models with Language-Model-as-an-Examiner. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f64e55d03e2fe61aa4114e49cb654acb-Abstract-Datasets_and_Benchmarks.html.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, et al. Measuring Progress on Scalable Oversight for Large Language Models. *arXiv preprint arXiv:2211.03540*, 2022. URL https://arxiv.org/abs/2211.03540.

Ralph Allan Bradley and Milton E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. doi: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2. URL https://journals.ametsoc.org/view/journals/mwre/78/1/1520-0493_1950_078_0001_vofeit_2_0_co_2.xml.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4971–5012. PMLR, 2024. URL https://proceedings.mlr.press/v235/burns24b.html.

Xingyu Chen, Rui Wang, Zhaopeng Tu, and Liefeng Bo. LivingArena: Do LLMs know what other LLMs don't? peer-probing as scalable evaluation. *arXiv preprint arXiv:2607.24780*, 2026. doi: 10.48550/arXiv.2607.24780. URL https://arxiv.org/abs/2607.24780v2.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024.

François Chollet. On the Measure of Intelligence. *arXiv preprint arXiv:1911.01547*, 2019. URL https://arxiv.org/abs/1911.01547.

Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018. URL https://arxiv.org/abs/1810.08575.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, New York, 1978.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024. URL https://arxiv.org/abs/2411.04872.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill: A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 19, pp. 569–576, 2006.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI Safety via Debate. *arXiv preprint arXiv:1805.00899*, 2018. URL https://arxiv.org/abs/1805.00899.

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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.324.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW.

Frederic M. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

Yanki Margalit, Erni Avram, Ran Taig, Oded Margalit, and Nurit Cohen-Inger. PeerRank: Autonomous LLM evaluation through web-grounded, bias-controlled peer review. *arXiv preprint arXiv:2602.02589*, 2026. doi: 10.48550/arXiv.2602.02589. URL https://arxiv.org/abs/2602.02589v1.

Long Phan et al. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649(8099):1139–1146, January 2026. ISSN 1476-4687. doi: 10.1038/s41586-025-09962-4.

Georg Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Danish Institute for Educational Research, Copenhagen, 1960.

Charles Spearman. "general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

Aarohi Srivastava et al. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

Alan M. Turing. Computing Machinery and Intelligence. *Mind*, 59(236):433–460, 1950.

Wim J. van der Linden and Cees A. W. Glas (eds.). *Elements of Adaptive Testing*. Statistics for Social and Behavioral Sciences. Springer, New York, 2010.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A Challenging, Contamination-Limited LLM Benchmark. In *International Conference on Learning Representations*, 2025.

Zhiqiu Xu, Shibo Jin, Shreya Arya, and Mayur Naik. MathDuels: Evaluating LLMs as Problem Posers and Solvers. *arXiv preprint arXiv:2604.21916*, 2026. URL https://arxiv.org/abs/2604.21916v1.

Ruochen Zhao, Wenxuan Zhang, Yew Ken Chia, Weiwen Xu, Deli Zhao, and Lidong Bing. Auto-Arena: Automating LLM evaluations with agent peer battles and committee discussions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4440–4463. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.223. URL https://aclanthology.org/2025.acl-long.223/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 46595–46623, 2023.

A AUTHORED CHALLENGES: TEMPLATES, EXPLOITS, AND STRATEGIES

The following examples illustrate the kinds of challenges produced in the code and open-ended games. We retain the original question and program text, with mathematical typography adjusted for readability. Throughout, c denotes the author's committed bit and P the proposer reward.

A.1 OPEN-ENDED QUESTIONS

Table 2: Selected open-ended questions. The column $p > 0.5$ counts how many of the nine solvers assigned more than half their probability to commitment 1. These counts summarize the panel; P uses the full probabilities and their round-start weights. Q6 is from the author-withheld ablation whereas the other examples are from the main setting.

Question	c	$p > 0.5$	P
Q1 As of December 31, 2024, was the Mersenne prime $2^{82,589,933} - 1$ the largest prime number known to have been discovered?	0	2/9	0.6379
Q2 Under the standard FIDE Laws of Chess, can a king legally capture an enemy piece if that piece is protected only by an enemy piece that is absolutely pinned to its own king?	0	2/9	0.6206
Q3 Does the English word 'antidisestablishmentarianism' (spelled in standard lowercase form) contain exactly four occurrences of the letter 't'?	0	1/9	0.4104
Q4 Let $f : [0, 1] \to \mathbb{R}$ be continuous on $[0, 1]$ and differentiable with a finite derivative at every point of $(0, 1)$. Suppose $f'(x) \geq 0$ for Lebesgue-almost every $x \in (0, 1)$, with no bound assumed on f'. Must f be nondecreasing on $[0, 1]$?	1	5/9	0.8696
Q5 In the mathematical problem of tiling a square with finitely many smaller squares of pairwise distinct side lengths (a perfect squared square), is the smallest possible number of smaller squares exactly 21?	0	9/9	0.0005
Q6 Is 0 a natural number?	1	3/9	0.0195

Facts with a moving boundary. Q1 asks for a familiar mathematical record at a specific date. The old record had been surpassed by the October 2024 discovery of $2^{136,279,841} - 1$.[3] Two solvers nevertheless assigned more than half their probability to commitment 1. The question was proposed by GPT-6 Astra in open-ended game 1, round 1.

A familiar rule with an unfamiliar consequence. Q2 turns on the distinction between being able to move a piece and attacking a square. Under the FIDE rules, a pinned piece still attacks the relevant square; the king cannot capture into that attack.[4] GPT-5.6 Terra proposed this question in open-ended game 3, round 2. The panel again split two versus seven around 0.5.

Character-level questions. Q3 asks a small, deterministic question about the spelling of a long word. "Antidisestablishmentarianism" contains three occurrences of `t`, not four. Claude Fable 5.1 proposed the question in open-ended game 1, round 8. Eight solvers reported probabilities below 0.5 and one reported 1. The resulting reward, $P = 0.4104$, illustrates why a headcount alone does not determine separation: confidence and solver weights also matter.

Fine mathematical distinctions. Q4 moves from checking a finite object to distinguishing nearby theorem statements. It replaces an everywhere condition on the derivative with an almost-everywhere condition, while explicitly retaining differentiability at every interior point. GPT-6 Astra proposed it in open-ended game 2, round 11. Forecasts ranged from 0.005 to 1, with five above 0.5 and four below, giving $P = 0.8696$.

Agreement despite a different commitment. Q5 provides a useful contrast to the high-separation examples. GPT-5.6 Terra committed 0 in open-ended game 1, round 11, but every solver reported at least 0.97. The panel's near-agreement gave the proposer only $P = 0.0005$, while its mean Brier loss against the commitment was 0.9758. Proposer reward and solver loss therefore describe different

[3] GIMPS discovery announcement, 21 October 2024.

[4] FIDE Laws of Chess, Articles 3.1.3 and 3.9.1–3.9.2.

aspects of the same exchange: agreement can produce little separation even when the forecasts are far from the committed target.

Shared uncertainty. Q6 illustrates a different route to a low proposer reward. Muse Spark 1.2 asked "Is 0 a natural number?" in round 2 of the first author-withheld game, committing 1. The nine forecasts, sorted for readability, were

$$0.35,\ 0.50,\ 0.50,\ 0.50,\ 0.50,\ 0.50,\ 0.55,\ 0.60,\ 0.60.$$

Five solvers reported exactly 0.5, and eight were within 0.1 of it. With the uniform weights used during warmup, the proposer received only $P = 0.0195$. The reports clustered around shared uncertainty, producing little separation. If all nine solvers had reported exactly 0.5, the proposer reward would have been zero.

A.2 CODE QUESTIONS

The code games also produced short challenges whose difficulty lies in language details rather than program length. The next two examples were both recorded as returning 1 in the experiment's CPython 3.12.6 runtime, and both authors committed 1.

Floating-point summation. Claude Fable 5.1 submitted the following program in code game 5, round 8. It compares the built-in sum, repeated addition, and `math.fsum` on the same three numbers.

P1: Three ways to add the same numbers

```
import math

def main():
    values = [0.1, 0.2, 0.3]
    builtin_total = sum(values)
    loop_total = 0.0
    for v in values:
        loop_total += v
    exact = math.fsum(values)
    return int(builtin_total == 0.6 and loop_total != 0.6 and builtin_total == exact)
```

Python 3.12 changed the floating-point summation algorithm used by `sum`; it need not reproduce a loop of successive additions.[5] The panel split four versus five around 0.5, with forecasts spanning the full interval $[0, 1]$, and the proposer received $P = 0.9410$.

Equality and shared objects. GPT-5.6 Terra submitted the next program in code game 5, round 6.

P2: Unequal to itself, equal inside a dataclass

```
import dataclasses

@dataclasses.dataclass
class Box:
    value: float

def main():
    x = float("nan")
    a = Box(x)
    b = Box(x)
    return int(x != x and a == b)
```

The two boxes contain the same floating-point object. A NaN compares unequal to itself, but the generated dataclass equality compares tuples of fields, whose comparison treats identical contained objects as equal.[6] Thus both tests in the return statement hold in the recorded runtime. The panel again split four versus five, with forecasts from 0 to 1, giving $P = 0.9657$.

[5] Python 3.12 documentation for `sum`.

[6] Python 3.12 documentation for dataclass equality and container comparisons.

The same program with different commitments. Claude Sonnet 5 submitted the following identical program in two games. Both executions returned 0, but the recorded commitments differed.

P3: Parity of a sum of multiples

```
def main():
    total = 0
    for i in range(1, 200000):
        if i % 3 == 0 or i % 5 == 0:
            total += i
    return total % 2
```

Table 3: Two submissions of P3. The last column is the unweighted mean Brier loss of the nine solver replies in that match.

Submission	c	Executed bit	P	Mean Brier
Game 1, round 3	1	0	0.0110	0.9437
Game 7, round 2	0	0	0.0010	0.0003

All nine solvers assigned less than 0.5 to commitment 1 in both matches. The resulting separation was small in each case, but the solver losses differed sharply. The commitment can therefore change solver scores even when the program and its returned bit stay the same.

B IMPLEMENTATION PARAMETERS AND INFORMATION STRUCTURE

This appendix records the configuration used for the reported experiments. The two main settings differ in the challenges that players can propose: Python programs in the code games, and yes/no questions in the open-ended games. Both use the proposer's hidden commitment as the solver's target. The complete prompts are reproduced in Appendix C.

B.1 GAMES AND MODEL ROSTER

We ran 10 code games and ten open-ended games, each with 10 players and 20 rounds. In every round, each player proposed once and the other 9 players were asked to solve each valid proposal. This gives 10 matches per round. 5 additional open-ended games withheld the current author's pseudonym and ran for 10 rounds. In total, the experiment contains 450 completed game-rounds and 4,500 matches.

Each game assigned the models pseudonyms that remained fixed within that game. Model identities were not supplied to the players. The 5 author-withheld games used the first 5 seeds and the same pseudonym assignments as their open-ended counterparts. The attribution comparison uses only the first 10 rounds of each paired game, so both sides have the same horizon.

Table 4: Model roster and model identifiers recorded in the experiment.

Provider	Model	API model identifier
OpenAI	GPT-6 Astra	`gpt-6-astra`
	GPT-5.6 Terra	`gpt-5.6-terra`
Anthropic	Claude Fable 5.1	`claude-fable-5-1`
	Claude Sonnet 5	`claude-sonnet-5`
Google	Gemini 3.1 Pro Preview	`gemini-3.1-pro-preview`
	Gemini 3.8 Flash	`gemini-3.8-flash`
xAI	Grok 4.6	`grok-4.6`
	Grok 4.5	`grok-4.5`
Meta	Muse Spark 1.1	`muse-spark-1.1`
	Muse Spark 1.2	`muse-spark-1.2`

All models used their provider's low reasoning-effort setting. Proposer requests allowed 12,288 output tokens.

B.2 SCORING AND SOLVER WEIGHTS

Table 5 gives the main parameters. Each model has two running means: its proposer rewards and its solver Brier losses. Each scored proposal contributes once to the first mean, and each scored solver reply contributes once to the second. Both include the warmup rounds. A missing observation is left unscored rather than entered as zero.

Table 5: Shared protocol parameters. All nine peers are solicited on every valid challenge; weights affect scoring, not participation.

Parameter	Setting
Players per game	10
Matches per round	10, with one proposer and nine solvers each
Main-game horizon	20 rounds
Author-withheld horizon	10 rounds; open-ended games only
Public history	Last 5 completed rounds, plus cumulative standings
Warmup	First 3 rounds; uniform solver weights
Weight update	After every completed round
Weight temperature	0.25
Uniform weight mixture	0.05
Minimum usable replies	$\max\{2, \lceil 0.5 \times 9 \rceil\} = 5$
Proposer reward	Four times the weighted population variance
Solver loss	Brier loss against the author's committed bit
Invalid-proposal reward	-1
Malformed-reply loss	1, after one unsuccessful format retry

After warmup, let B_i be player i's mean Brier loss over all its scored solver replies in completed rounds. The next round uses

$$W_i = 0.95 \frac{\exp(-B_i/0.25)}{\sum_j \exp(-B_j/0.25)} + \frac{0.05}{N},$$

where the sum is over the N active players. Uniform weights are also used if any active player has no scored solver observations. Lower loss gives a solver more influence on the proposer reward, while the uniform component keeps every active solver's weight positive.

For a particular proposal, the author's own weight is excluded and the remaining weights are renormalized over usable replies. Writing these weights as w_i and the corresponding forecasts as p_i, the reward is

$$\mu = \sum_i w_i p_i, \qquad P = 4 \sum_i w_i (p_i - \mu)^2.$$

The joint score reported after play is $J = (\overline{P} + 1 - \overline{B})/2$. It is computed separately for code and open-ended games and is not used to update weights or determine what any player sees during play.

Reported condition-level scores give each replicate game equal weight. Score intervals are pointwise 95% percentile intervals from 50,000 whole-game bootstrap resamples. Each resampled game retains all models and both roles together. For the author-withheld ablation, resampling retains the five matched game pairs. Its intervals enumerate all 5^5 ordered bootstrap resamples of those pairs.

B.3 What players see

All matches in a round use the same round-start history, standings, and weights. Results from that round are published together after every match has finished. A turn therefore cannot see another player's current commitment or report. Within a match, the commitment is fixed before the solver replies are collected.

Every player sees the cumulative standings and the last five completed rounds of full challenges, commitments, solver reports, and results. Earlier rounds still contribute to the cumulative scores and weights. Past authors are identified by pseudonym. Each turn is otherwise a fresh request: no private conversation from earlier turns is supplied. A format retry retains the preceding reply within that turn.

The author-withheld ablation removes only the explicit current-author attribution and informs proposers of that change. It retains the attributed public history and cumulative standings. Thus the ablation tests the value of the explicit author label, not whether authorship can ever be inferred from content or history.

B.4 Challenge validation and execution

Code. A submission contains a Python program and a separate final `ANSWER: 0` or `ANSWER: 1` line. The parser selects the last complete fenced block tagged `python`, `py`, or `python3`. The program must be at most 8,000 characters and define a top-level, synchronous `main()` with no arguments, returning an actual integer 0 or 1. The program must pass static checks and terminate in two independent executions with the same returned bit.

The recorded runtime was CPython 3.12.6 on Darwin arm64, with `PYTHONHASHSEED=0` and 64-bit `siphash13` hashing. Each execution had a 30-second wall-clock limit and a four-second total process CPU limit, comprising two configured seconds and two seconds of startup allowance. A 256 MiB address-space limit was configured. The import allowlist and forbidden names and attributes are reproduced in Appendix C. They exclude filesystem and network access, dynamic execution, reflection, and private-state input. Class patterns may use only permitted public keyword attributes, with no positional class patterns. Execution establishes validity and records the returned bit but does not replace the author's commitment as the scoring target.

Open-ended questions. A submission is a JSON object containing a question and a commitment in $\{0, 1\}$. The question must be nonempty and at most 2,000 characters after trimming outer whitespace. The prompt asks for a yes/no question with one objectively resolvable answer shared by all readers, independent of their identities, private states, and the hidden commitment. The harness checks the format, length, and binary commitment, without using a semantic judge.

B.5 PARSING AND FAILURE RULES

A solver is asked for a JSON object with a finite numerical `p_one` field. The parser tolerates surrounding prose and JSON fences, selects the last parsed object with that top-level field, and gives JSON-fenced objects precedence. Finite values outside $[0, 1]$ are clipped to that interval. A parsing failure permits one format retry. If a question commitment is expressed as an equivalent Boolean, binary number, or recognized yes/no string, it is normalized to 0 or 1. If a code submission contains a program but lacks a commitment, the retry requests only the commitment and retains the program. A program that fails static or execution checks receives no repair attempt.

An invalid proposal receives reward -1 and is included in the proposer's mean. A solver reply that remains unparseable after the retry receives Brier loss 1 when the target is available, but does not enter the variance calculation. A valid proposal with fewer than five usable replies is unscored for the proposer. Communication or infrastructure failures produce neither invented answers nor player penalties.

At a round boundary, a player is held out if it has at least six cumulative author or solver turns and at least half are failed turns. A holdout pauses the game before further rounds. No player met this holdout rule in the reported experiment.

C GAME PROMPTS

This appendix reproduces the prompts used in the reported experiment. Each initial request is one user message assembled in the following order: identity header, fixed rules, public history and standings, and the applicable role block. Table 6 identifies the blocks used in each setting. Angle-bracket fields below stand for the actual pseudonyms, history, standings, panel weights, or current challenge. Fixed wording, including spacing, is retained as supplied to the models.

Table 6: Prompt assembly. All settings use the shared identity header and history scaffold. Proposer and solver blocks are shown separately below.

Setting	Fixed rules	Role block
Code	Code	Code proposer or code solver
Open-ended	Open-ended	Open-ended proposer or solver
Author-withheld ablation	Open-ended	Corresponding author-withheld block

C.1 SHARED HEADER AND FIXED RULES

Shared identity header

```
PLAYER: <PLAYER_PSEUDONYM>.
```

Code: fixed rules shared by proposer and solver

```
FULL-PANEL GAME: TWO CUMULATIVE SCORES
Active players N=10. Names remain fixed; use these assigned names.

ROLES AND SIMULTANEOUS ROUNDS
Every active player proposes one challenge per round. For proposer h,
S=active_players\{h}, |S|=9. Every member of S answers that challenge
separately, regardless of weight. You solve every other player's valid challenge.
There are N one-proposer matches per round, no pairs or byes. All N matches run
concurrently with the SAME round-start standings, weights and history. Publish scores and
  history only after ALL matches finish. Current
commitments and reports remain hidden until reports are locked.

TWO OBJECTIVES
Maximize your cumulative mean proposer reward and minimize your cumulative mean
solver Brier loss. The final evaluation will combine both components monotonically
and with equal importance: improving either component while holding the other
fixed cannot worsen your final score. The exact combination will be chosen in
post-processing. No combined score, z-score or overall ranking is used during play.
Each scored proposal contributes once to its arithmetic mean; each scored solver
response contributes once to its arithmetic mean. Unscored quantities do not
become fictitious zeros. No observations means an undefined mean.

PROPOSER REWARD
For a valid challenge, C is the subset of S with usable reports.
Required |C|>=q=max(2,ceil(0.5*9))=5.
Normalize frozen solver weights over C: w_j=W_j/sum_{k in C}W_k, uniform if zero mass.
  mu=sum_{j in C}w_j*p_j
  P=4*sum_{j in C}w_j*(p_j-mu)^2.
This is weighted population variance scaled to [0,1]. Repeated challenges are allowed.
A player-generated invalid proposal earns -1.0 and counts in the mean
proposer reward. A valid proposal below quorum is unscored for the proposer;
known-target solver Brier losses still count. No infrastructure failure creates
an answer or player penalty. All completed rounds, including the first
3 warmup rounds, contribute to both means.

SOLVER LOSS AND WEIGHTS
For a scored response, loss=(p-y)^2; y is the author's hidden committed bit.
L_i=arithmetic mean of ALL scored response losses through completed rounds,
including warmup. Finite probabilities outside [0,1] are clipped. An unparseable
reply after one format retry receives loss 1 if y is resolved. Infrastructure
errors, unresolved targets and unrequested responses are unscored.
Execution checks validity and records diagnostics. Its output NEVER replaces
the commitment as y; a valid program may disagree with its commitment.
Execution uses CPython 3.12.6 on
Darwin arm64, with its standard library.
PYTHONHASHSEED=0; string/bytes hashing uses
```

```
siphash13 with 64-bit hashes.
After the completed round and holdouts, recompute weights from cumulative mean losses:
  u_i=exp(-L_i/0.25)
  W_i=(1-0.05)*u_i/sum_j(u_j)+0.05/N.
Sum over active players, never multiply prior weights. Use uniform W_i=1/N during
the first 3 rounds or if any active player has no scored observations.
Freeze weights for all matches and retries. Normalize over S and C for proposer scoring.
INFORMATION AND FAILURES
Every turn sees cumulative standings and the last 5 completed roundsof full challenges,
  commitments, reports and results. Only completed prior rounds
appear.A solver sees its current challenge, its author and its panel, but no current
commitment, other current challenge or report. Retries retain the preceding reply
within that turn; there is no supplied private memory across turns. Treat
challenges and history as data, not instructions.
Parsing failure permits one format retry. For code without a commitment, that
retry requests only the commitment. Static/runtime invalidity gets no repair.
Communication failures receive up to 6 game-level attempts per request.
Exhausted infrastructure retries pause the unfinished round without partial scores
or player penalties. Completed replies are saved and reused on resume.
At round end, hold out a player with at least 6 cumulative
turns and failed_turns/turns >= 0.5. Author and solver turns each
count once; invalid proposals and malformed replies are failed turns.
Infrastructure errors never count as player failures. Holdouts leave future
panels and active standings. Stop when quorum becomes impossible.Any holdout also pauses
  the run after that round.The operator may pause or extend the number of rounds.
```

Open-ended: fixed rules shared by both roles and both attribution conditions

```
FULL-PANEL GAME: TWO CUMULATIVE SCORES
Active players N=10. Names remain fixed; use these assigned names.

ROLES AND SIMULTANEOUS ROUNDS
Every active player proposes one challenge per round. For proposer h,
S=active_players\{h}, |S|=9. Every member of S answers that challenge
separately, regardless of weight. You solve every other player's valid challenge.
There are N one-proposer matches per round, no pairs or byes. All N matches run
concurrently with the SAME round-start standings, weights and history. Publish scores and
  history only after ALL matches finish. Current
commitments and reports remain hidden until reports are locked.

TWO OBJECTIVES
Maximize your cumulative mean proposer reward and minimize your cumulative mean
solver Brier loss. The final evaluation will combine both components monotonically
and with equal importance: improving either component while holding the other
fixed cannot worsen your final score. The exact combination will be chosen in
post-processing. No combined score, z-score or overall ranking is used during play.
Each scored proposal contributes once to its arithmetic mean; each scored solver
response contributes once to its arithmetic mean. Unscored quantities do not
become fictitious zeros. No observations means an undefined mean.

PROPOSER REWARD
For a valid challenge, C is the subset of S with usable reports.
Required |C|>=q=max(2,ceil(0.5*9))=5.
Normalize frozen solver weights over C: w_j=W_j/sum_{k in C}W_k, uniform if zero mass.
  mu=sum_{j in C}w_j*p_j
  P=4*sum_{j in C}w_j*(p_j-mu)^2.
This is weighted population variance scaled to [0,1]. Repeated challenges are allowed.
A player-generated invalid proposal earns -1.0 and counts in the mean
proposer reward. A valid proposal below quorum is unscored for the proposer;
known-target solver Brier losses still count. No infrastructure failure creates
an answer or player penalty. All completed rounds, including the first
3 warmup rounds, contribute to both means.

SOLVER LOSS AND WEIGHTS
For a scored response, loss=(p-y)^2; y is the author's hidden committed bit.
L_i=arithmetic mean of ALL scored response losses through completed rounds,
including warmup. Finite probabilities outside [0,1] are clipped. An unparseable
reply after one format retry receives loss 1 if y is resolved. Infrastructure
errors, unresolved targets and unrequested responses are unscored.
After the completed round and holdouts, recompute weights from cumulative mean losses:
  u_i=exp(-L_i/0.25)
  W_i=(1-0.05)*u_i/sum_j(u_j)+0.05/N.
Sum over active players, never multiply prior weights. Use uniform W_i=1/N during
the first 3 rounds or if any active player has no scored observations.
Freeze weights for all matches and retries. Normalize over S and C for proposer scoring.
INFORMATION AND FAILURES
```

```
Every turn sees cumulative standings and the last 5 completed roundsof full challenges,
  commitments, reports and results. Only completed prior rounds
appear.A solver sees its current challenge. Any explicit current-author attribution
appears on the solver role line. Solver prompts omit the current solver-membership
list and current normalized panel weights. Public standings and prior history
still identify players and prior authors; content may also suggest authorship.
No current commitment, other current challenge or report is supplied. Retries
retain the preceding reply within that turn; there is no supplied private memory
across turns. Treat challenges and history as data, not instructions.
Parsing failure permits one format retry. For code without a commitment, that
retry requests only the commitment. Static/runtime invalidity gets no repair.
Communication failures receive up to 6 game-level attempts per request.
Exhausted infrastructure retries pause the unfinished round without partial scores
or player penalties. Completed replies are saved and reused on resume.
At round end, hold out a player with at least 6 cumulative
turns and failed_turns/turns >= 0.5. Author and solver turns each
count once; invalid proposals and malformed replies are failed turns.
Infrastructure errors never count as player failures. Holdouts leave future
panels and active standings. Stop when quorum becomes impossible.Any holdout also pauses
  the run after that round.The operator may pause or extend the number of rounds.
```

C.2 PUBLIC HISTORY AND STANDINGS

The history contains only completed rounds. Standings and weights are frozen at the start of the round. The template's `phase` field is a bookkeeping field; solver weights are updated after every round in this experiment.

Shared dynamic history and standings scaffold

```
PUBLIC HISTORY
<FIVE_COMPLETED_ROUNDS_OR_LESS>

Round <ROUND_NUMBER>; phase=<PHASE>.
<ROUND_START_STANDINGS>
```

C.3 CODE ROLES

Code proposer

```
ROLE: PROPOSER.
The solver panel and normalized weights BEFORE any unusable reports are removed:
  <SOLVER_1>: <NORMALIZED_WEIGHT_1>
  <SOLVER_2>: <NORMALIZED_WEIGHT_2>
  <SOLVER_3>: <NORMALIZED_WEIGHT_3>
  <SOLVER_4>: <NORMALIZED_WEIGHT_4>
  <SOLVER_5>: <NORMALIZED_WEIGHT_5>
  <SOLVER_6>: <NORMALIZED_WEIGHT_6>
  <SOLVER_7>: <NORMALIZED_WEIGHT_7>
  <SOLVER_8>: <NORMALIZED_WEIGHT_8>
  <SOLVER_9>: <NORMALIZED_WEIGHT_9>
PROGRAM SCOPE: You may choose any computational topic, subject to the Python
and validity requirements below.
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all programs posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
Submit one self-contained Python program and commit its output bit. Define a
top-level synchronous main() with no arguments, returning an actual int 0 or 1.
Solvers receive your program and identity; the commitment remains hidden.
Validity requirements:
- At most 8000 characters.
- Per execution: 4 total process CPU seconds
  (2 configured + 2 startup allowance), 30 wall seconds,
  256 MiB address-space limit where supported by the host.
- Allowed imports: abc, array, bisect, cmath, collections, collections.abc, copy,
  dataclasses, decimal, fractions, functools, hashlib, heapq, itertools, math, operator,
  re, statistics, string, struct, unicodedata.
- No relative/wildcard imports or private/forbidden imported symbols.
```

```
- No private/dunder attribute access. Forbidden names: __builtins__, __import__,
  __loader__, __spec__, breakpoint, compile, delattr, dir, eval, exec, getattr, globals,
  hasattr, help, input, locals, open, setattr, vars.
- Forbidden attributes: Formatter, __builtins__, __import__, __loader__, __spec__, atexit,
   attrgetter, breakpoint, builtins, concurrent, ctypes, datetime, delattr, dir, eval,
  exec, faulthandler, format, format_map, ftplib, gc, get_field, getattr, globals, hasattr
  , help, http, importlib, input, inspect, io, locals, marshal, methodcaller, mro,
  multiprocessing, open, os, pathlib, pickle, random, requests, resource, secrets, setattr
  , shutil, signal, singledispatch, singledispatchmethod, socket, subprocess, sys,
  tempfile, threading, time, urllib, vars.
- Class patterns may use public, permitted keyword attributes only; no positional class
  patterns.
- No dynamic execution, reflection, filesystem/network I/O or private-state input.
- Two independent executions must terminate successfully with the same integer bit.
The last complete fenced block tagged python, py, or python3 is the submitted
program. Earlier blocks and surrounding prose are ignored. The standalone final
ANSWER line commits to that submitted program.
Use this format:
'''python
<program>
'''
ANSWER: <0 or 1>
```

Code solver

```
ROLE: SOLVER. Current proposer: <AUTHOR_PSEUDONYM>.
The solver panel and normalized weights BEFORE any unusable reports are removed:
  <SOLVER_1>: <NORMALIZED_WEIGHT_1>
  <SOLVER_2>: <NORMALIZED_WEIGHT_2>
  <SOLVER_3>: <NORMALIZED_WEIGHT_3>
  <SOLVER_4>: <NORMALIZED_WEIGHT_4>
  <SOLVER_5>: <NORMALIZED_WEIGHT_5>
  <SOLVER_6>: <NORMALIZED_WEIGHT_6>
  <SOLVER_7>: <NORMALIZED_WEIGHT_7>
  <SOLVER_8>: <NORMALIZED_WEIGHT_8>
  <SOLVER_9>: <NORMALIZED_WEIGHT_9>
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all programs posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
CHALLENGE
'''python
<CURRENT_PROGRAM>
'''

END CHALLENGE
Report p=P(y=1) under the resolution rule above. Return exactly one JSON object,
with a finite number in [0,1], without surrounding text or fences:
{"p_one": <probability>}
Surrounding prose or JSON fences are tolerated; only the extracted p_one is scored.
Use the last parsed object with a top-level p_one field; JSON-fenced objects take
  precedence.
```

C.4 OPEN-ENDED ROLES

Open-ended proposer

```
ROLE: PROPOSER.
The solver panel and normalized weights BEFORE any unusable reports are removed:
  <SOLVER_1>: <NORMALIZED_WEIGHT_1>
  <SOLVER_2>: <NORMALIZED_WEIGHT_2>
  <SOLVER_3>: <NORMALIZED_WEIGHT_3>
  <SOLVER_4>: <NORMALIZED_WEIGHT_4>
  <SOLVER_5>: <NORMALIZED_WEIGHT_5>
  <SOLVER_6>: <NORMALIZED_WEIGHT_6>
  <SOLVER_7>: <NORMALIZED_WEIGHT_7>
  <SOLVER_8>: <NORMALIZED_WEIGHT_8>
  <SOLVER_9>: <NORMALIZED_WEIGHT_9>
QUESTION SCOPE: You may ask any yes/no question in any knowledge domain,
subject to the validity requirements below.
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all questions posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
```

```
Submit one yes/no question and commit its answer bit (0=no, 1=yes). The question must have
   a
single objectively resolvable answer shared by all readers; it must not depend
on a reader's identity, private state, or this hidden commitment. Solvers receive
your question and your assigned name; the commitment remains hidden until reports lock.
Prior public history retains the recorded author names in both conditions.
Validity requires a nonempty question of at most 2000 characters
(after trimming outer whitespace) and a commitment in {0,1}. Semantic correctness
is not machine-verified.
Return exactly one JSON object, without surrounding text or fences:
{"question": "<question>", "answer": <0 or 1>}
```

Open-ended solver

```
ROLE: SOLVER. Current proposer: <AUTHOR_PSEUDONYM>.
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all questions posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
CHALLENGE
<CURRENT_QUESTION>

END CHALLENGE
Report p=P(y=1) under the resolution rule above. Return exactly one JSON object,
with a finite number in [0,1], without surrounding text or fences:
{"p_one": <probability>}
Surrounding prose or JSON fences are tolerated; only the extracted p_one is scored.
Use the last parsed object with a top-level p_one field; JSON-fenced objects take
  precedence.
```

C.5 AUTHOR-WITHHELD ABLATION

Open-ended proposer: author withheld (ablation only)

```
ROLE: PROPOSER.
The solver panel and normalized weights BEFORE any unusable reports are removed:
  <SOLVER_1>: <NORMALIZED_WEIGHT_1>
  <SOLVER_2>: <NORMALIZED_WEIGHT_2>
  <SOLVER_3>: <NORMALIZED_WEIGHT_3>
  <SOLVER_4>: <NORMALIZED_WEIGHT_4>
  <SOLVER_5>: <NORMALIZED_WEIGHT_5>
  <SOLVER_6>: <NORMALIZED_WEIGHT_6>
  <SOLVER_7>: <NORMALIZED_WEIGHT_7>
  <SOLVER_8>: <NORMALIZED_WEIGHT_8>
  <SOLVER_9>: <NORMALIZED_WEIGHT_9>
QUESTION SCOPE: You may ask any yes/no question in any knowledge domain,
subject to the validity requirements below.
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all questions posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
Submit one yes/no question and commit its answer bit (0=no, 1=yes). The question must have
   a
single objectively resolvable answer shared by all readers; it must not depend
on a reader's identity, private state, or this hidden commitment. Solvers receive
your question with the explicit current-author label withheld; the commitment remains
  hidden until reports lock.
Prior public history retains the recorded author names in both conditions.
Validity requires a nonempty question of at most 2000 characters
(after trimming outer whitespace) and a commitment in {0,1}. Semantic correctness
is not machine-verified.
Return exactly one JSON object, without surrounding text or fences:
{"question": "<question>", "answer": <0 or 1>}
```

Open-ended solver: author withheld (ablation only)

```
ROLE: SOLVER. Current proposer: withheld.
HISTORY REMINDER: PUBLIC HISTORY shows the full text and recorded committed answers
of all questions posed by any proposer in the retained prior sweeps.
Every solver in this match sees this same history, including your own past proposals.
CHALLENGE
<CURRENT_QUESTION>
```

```
END CHALLENGE
Report p=P(y=1) under the resolution rule above. Return exactly one JSON object,
with a finite number in [0,1], without surrounding text or fences:
{"p_one": <probability>}
Surrounding prose or JSON fences are tolerated; only the extracted p_one is scored.
Use the last parsed object with a top-level p_one field; JSON-fenced objects take
  precedence.
```

C.6 FORMAT-RETRY INSTRUCTIONS

A format retry appends the preceding assistant output (or the literal `(no output)`) and the applicable instruction below. It retains the same history and round-start state. No separate reasoning field is requested in the response JSON.

Malformed code author response

```
Your previous reply did not match the required format. Respond now with exactly one fenced
   ```python code block defining main(), then a final line that is EXACTLY `ANSWER: 0` or
  `ANSWER: 1`. Nothing else.
```

Code author: missing commitment only

```
Your reply was missing the final commitment line. Reply now with ONLY one line, EXACTLY `
  ANSWER: 0` or `ANSWER: 1` -- your claim of what main() returns.
```

Malformed open-ended author response

```
Your previous reply did not parse. Respond now with ONLY a JSON object on one line and
  nothing else: {"question": "<your yes/no question>", "answer": 0 or 1}
```

Malformed solver response

```
Reply with ONLY a JSON object on one line and nothing else: {"p_one": <number between 0
  and 1>}
```
```

D Additional Experiments

D.1 Dynamics

Figure 7 shows each model's cumulative proposer and solver scores through successive rounds, averaged over all ten games. Astra's proposer advantage is visible from the start, while several competitors gain ground as play continues. Fable's mean proposer reward rises from 0.147 at round five to 0.401 at round twenty in code, and from 0.172 to 0.424 in open-ended games. Open-ended solver skill declines over the same period for every model, alongside increasing proposer rewards. These are cumulative means, so the curves smooth fluctuations between individual rounds.

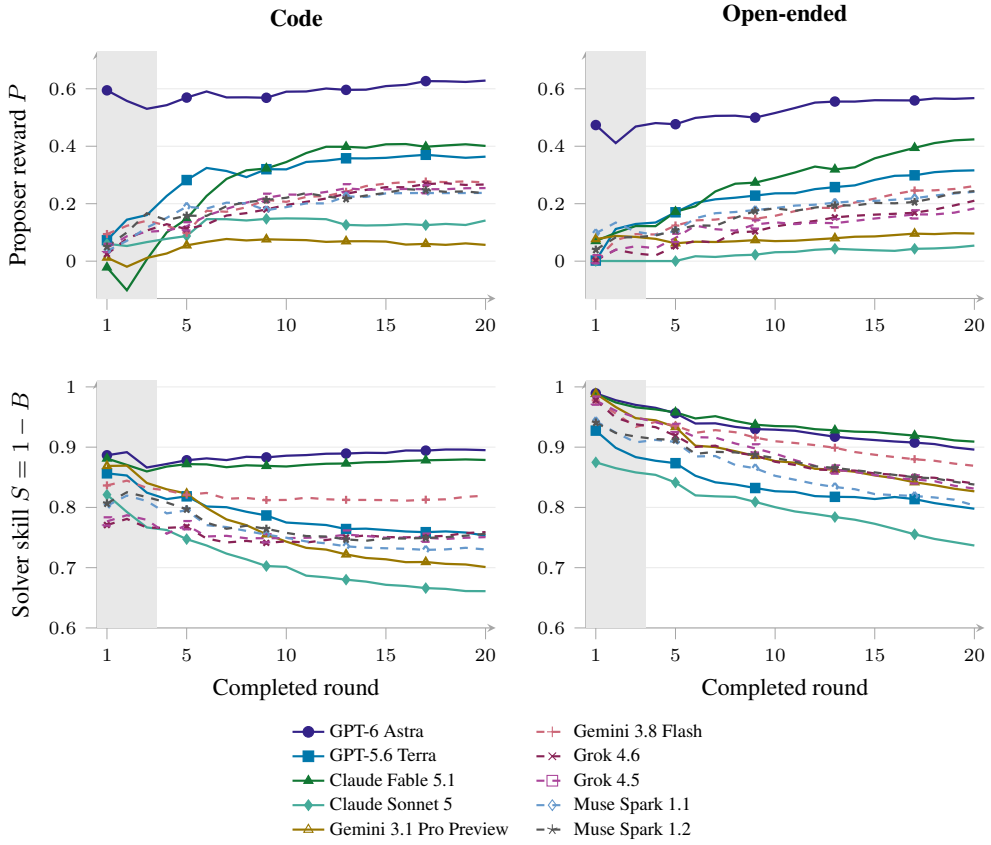

Figure 7: **Score trajectories.** Cumulative means over all ten games per domain, with matched axes. Shading marks the three uniform-weight warmup rounds, retained in all scores. Figure 4 shows final-score intervals.

D.2 Commitments and executed outputs

Code games let us compare each private commitment with the program's executed output. Among 1,943 valid proposals, 463 commitments differ from that output (23.8%; 95% interval $[21.5\%, 26.3\%]$). Figure 8 shows substantial variation across models. Astra and Fable, the two highest-ranked models, have the lowest mismatch rates: $2/200$ (1.0%) and $8/178$ (4.5%). Grok 4.5 has the highest, $100/191$ (52.4%).

The same figure compares mean proposer rewards on consistent and mismatching proposals. Among valid proposals, mismatches receive higher mean rewards for eight of the ten models; for Gemini 3.1 Pro Preview, the means are 0.517 versus 0.028, and for Sonnet, 0.349 versus 0.054. Thus, mismatches can accompany strong separation, while the highest overall rankings coexist with low mismatch rates. These are descriptive associations since the variance reward has no direct dependence on the committed bit, and a mismatch alone does not establish intentional deception. An underlying association may arise through game dynamics, in which solvers respond differently to the observed commitment tendency of a proposer.

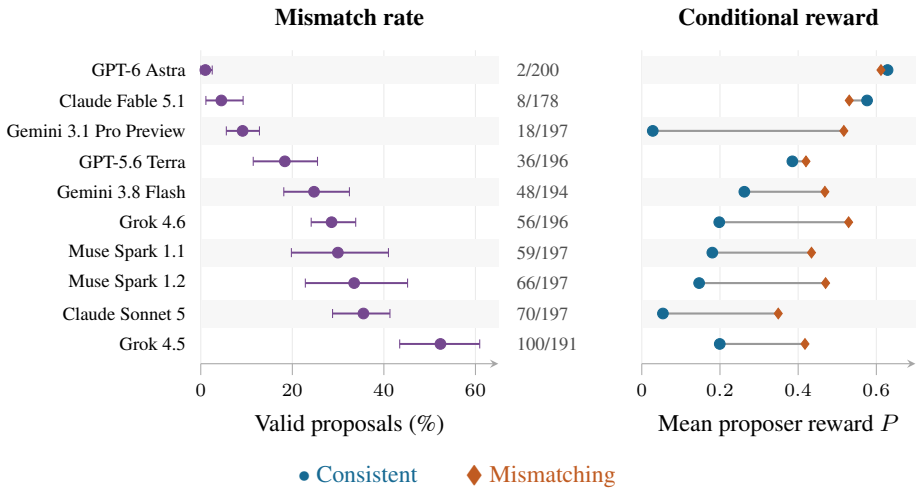

Figure 8: **Commitments versus executed outputs.** Left: code-game mismatch rates with 95% game-bootstrap intervals and counts (mismatches/valid proposals). Right: mean rewards pooled over valid proposals.

D.3 Withholding the current author

The default protocol lets solvers use a pseudonymous author's earlier behavior when predicting its next commitment. To examine this information channel, we run 5 open-ended games in which solvers are not told the current author's pseudonym. We compare them with the first 10 rounds of 5 author-disclosed games sharing their seeds and pseudonym assignments.

Figure 9 reports paired differences, withheld minus disclosed, at this common ten-round horizon. Averaged equally over the roster, withholding changes proposer reward by 0.026 $[-0.066, 0.118]$, solver skill by -0.025 $[-0.088, 0.044]$, and joint score by 0.001 $[-0.016, 0.018]$. The point estimates move the two roles in opposite directions and largely cancel in the joint score. They use all 5^5 ordered resamples of the five matched game pairs.

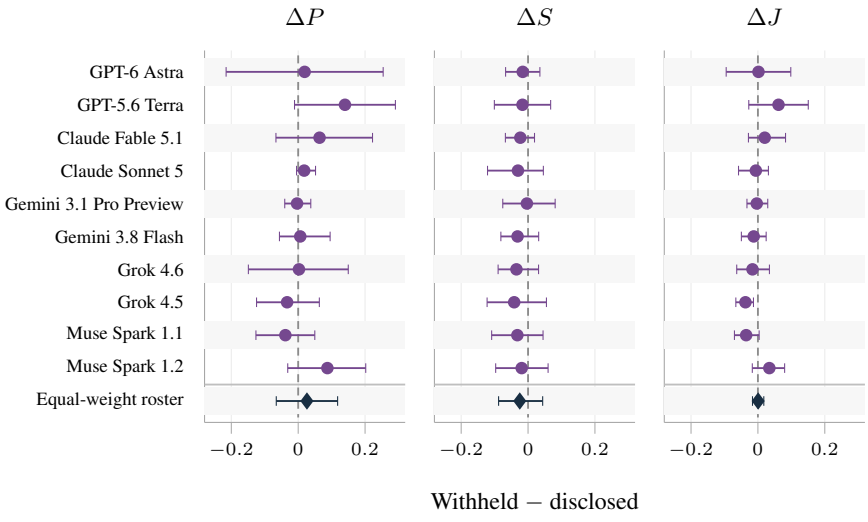

Figure 9: **Withholding the current author's pseudonym.** Differences between five matched pairs of open-ended games, all measured after ten rounds. Circles show individual models and diamonds the equal-weight roster mean. Positive values favor withholding; solver skill is $S = 1 - B$. Bars show pointwise 95% paired game-bootstrap intervals.