

# Predicting LLM Safety Before Release by Simulating Deployment

Marcus Williams\*, Hannah Sheahan\*, Cameron Raymond\*, Tomek Korbak\*,  
Deng Pan, Peilin Yang, Leon Maksin, Ningyi Xie, Phillip Guo, Ian Kivlichan,  
Micah Carroll\*<sup>†</sup>

OpenAI

## Abstract

Pre-deployment safety evaluations aim to inform the downstream risks of releasing a new AI model. Yet most evaluations provide limited evidence about how often undesired model behavior will occur in deployment: they generally have insufficient coverage, are unrepresentative, and are generally recognizable as tests. To address these concerns, we study a simple way to simulate a model deployment: starting from de-identified conversations from a previous model deployment, we hold fixed the initial conversation prefix and regenerate the next response using a candidate model. The resulting responses can then both be audited for novel misalignments and used to estimate the prevalence of model misbehavior before deployment. We evaluate deployment simulation across four GPT-5-series deployments, using outcome-blinded predictions for GPT-5.4 and retrospective analyses of three earlier releases. We find that deployment simulation produces informative estimates of post-deployment misbehavior rates and outperforms baselines based on adversarially selected production data; its evaluation-awareness point estimates were also much closer to production traffic than those from traditional evaluations. We also identify the realism of tool resampling as a central challenge for further improving predictions and share results suggesting that this challenge is surmountable even in complex tool-use settings. Finally, we show that deployment simulation can be seeded from public chat datasets and remain informative about production misbehavior rates, suggesting a path for external researchers to run deployment-grounded evaluations without access to private production logs. Overall, deployment simulation helps evaluators forecast how language models will behave in the real world and supports more quantitative assessment of deployment risk.

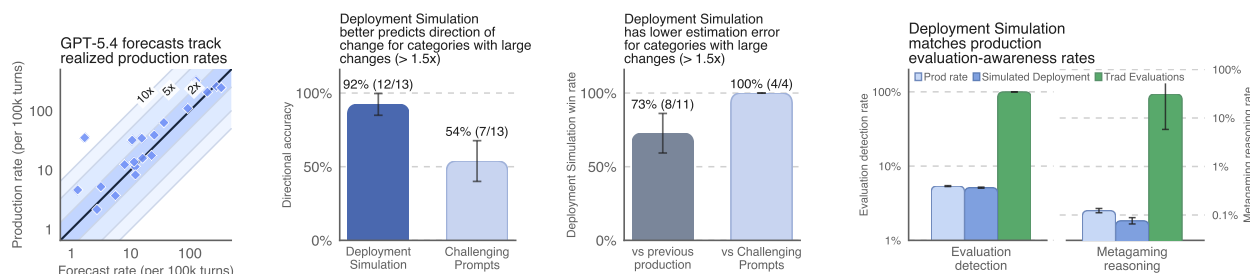


Figure 1: Our deployment simulation pipeline can predict production misbehavior rates before release, substantially outperforms baselines for categories whose production frequencies changed by at least  $1.5\times$ , and—unlike traditional evaluations—has similar evaluation-awareness rates to production.

<sup>†</sup>Corresponding author: [mdc@openai.com](mailto:mdc@openai.com). \*Core contributors.

# 1 Introduction

Pre-deployment evaluations used across the industry generally consist of a mix of synthetic, manually written, or production prompts intentionally selected to be difficult, high severity, or adversarial. These evaluations generally serve two intertwined goals: assessing how the model responds when stress-tested in situations that have a very small chance of occurring in production traffic, and helping characterize model misbehaviors, including novel undesired behaviors and their deployment-time frequencies.

While traditional evaluations remain crucial for stress-testing and capturing tail risks, we show how deployment simulations can be a significant improvement over traditional evaluations in pursuit of the second goal by sidestepping three central limitations of traditional evaluations. First, pre-deployment evaluation suites may be too narrow: they may not sufficiently cover the full breadth of uses that could induce misbehavior with a new model. Second, current evaluations often measure behavior on prompts drawn from a deliberately distorted slice of possible inputs: handcrafted prompts, synthetic scenarios, or adversarial examples selected because they are likely to elicit failures. Finally, models sometimes infer that they are being evaluated, leading them to behave differently, which may distort our risk assessments [1]. As a result of these three factors, traditional evaluation results may not be well-correlated with model behavior in deployment. They may also fail to surface new misbehaviors introduced by the candidate model, leaving developers surprised by safety failures only once it has already been deployed.

By simulating the candidate model’s deployment as realistically as possible, we aim to enable pre-deployment risk assessment that alleviates concerns about coverage, unrepresentativeness, and evaluation awareness. The method is simple: starting from de-identified production conversations from a recent deployment, we hold fixed the initial conversation prefix—which may include multiple user and assistant turns—and then resample the next model response using the “candidate model” to be released.

Deployment simulation avoids the unrepresentativeness problem of traditional evaluations: because we use a representative distribution of production conversations, the simulated conversations are also representative of the contexts in which we expect the model to operate in at deployment time. This way, deployment simulation largely decomposes coverage into a sample-size problem (that is, simulating enough data points) and an auditing problem (that is, having a method with sufficient sensitivity to find misbehavior of interest in the simulated data). Finally, by evaluating models on production contexts, we side-step most evaluation awareness cues.

In practice, we use the simulated conversations in two ways before deployment: first, to identify novel forms of misalignment introduced by the candidate model; and second, to estimate deployment-time misbehavior prevalence. The incidence of a given misbehavior in the simulated distribution can be treated directly as a prediction of deployment-time prevalence. Using the same measurement procedure after release also allows these predictions to be validated against the observed realized rates in production after release.

We show that deployment simulation generally outperforms traditional evaluations (Section 3.1), demonstrate its benefits for evaluation awareness (Section 3.2), and show promise in addressing the challenges of resampling agentic settings with heavy tool use (Section 3.3). We also investigate whether public chat data can make deployment-grounded auditing more accessible to external researchers (Section 3.4). Our findings suggest that deployment simulation can support accountable pre-deployment risk assessment: it enables developers to make predictions not only about what

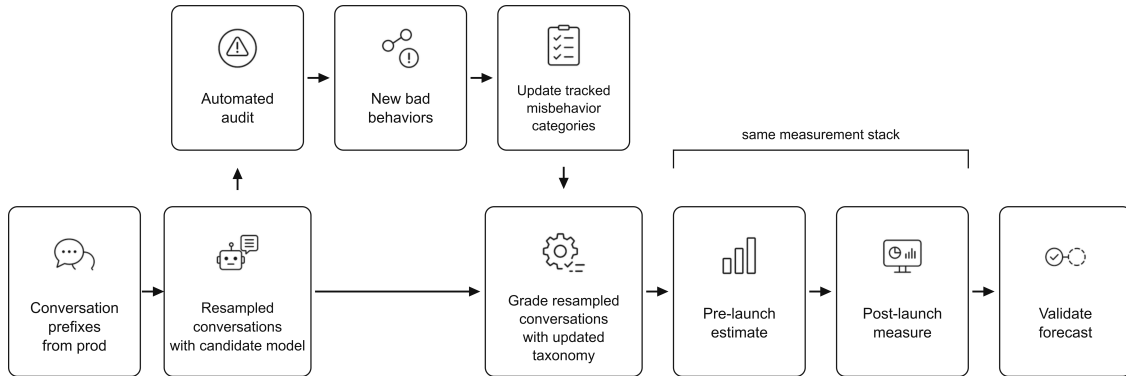


Figure 2: **Our deployment simulation pipeline.** We take representative conversation prefixes from recent production traffic and resample the next assistant response with a model yet to be released. These resampled conversations are audited for potential new misbehaviors, and then evaluated to produce pre-release prevalence estimates of misbehavior. After release, the same measurement stack is re-run on production traffic to validate the predictions.

might go wrong, but also about how often we expect each known misbehavior to occur in real usage. Sharing such predictions and validations in system cards can also make risk assessments more transparent to external stakeholders.

Our main findings are:

- **Simulated deployments improve forecasts of deployment misbehavior.** Across GPT-5-series Thinking model releases, our simulated deployments were substantially more informative about deployment-time misbehavior changes and rates than current pre-deployment evaluations.
- **Simulated deployments can surface novel misalignments before release.** Simulated outputs can be audited to find novel forms of misalignment. We show that this audit would have surfaced “calculator hacking” (a new form of reward hacking in GPT-5.1) prior to deployment.
- **Simulated deployments substantially reduce evaluation awareness.** Deployment simulation with real prompts induces far lower evaluation-awareness than traditional evaluations, with rates similar to real production traffic—which is reassuring for the validity of our risk estimates.
- **Realistic tool use simulation is a challenge, but can be mitigated even in agentic settings.** Realistic simulation is difficult when tool use depends on external state. Tool simulation show promise in addressing this challenge, even for simulating internal agentic deployments.
- **Our approach also remains informative with public chat datasets, enabling external auditing.** Public chat datasets such as WildChat provide a public analogue to production data. We found that WildChat yielded weaker—but still informative—pre-deployment risk predictions. This can empower external parties to perform independent audits across providers.

## 2 Method

Below, we describe how to simulate the deployment of a candidate model  $X$  using production data from deployed model  $Y$ . Following prior work that treats changes to deployed learning systems as counterfactual interventions [2], we hold historical context fixed while changing the model that produces the next response.

**Step 1: Sample representative production prefixes.** From recent production conversations generated with model  $Y$ , we uniformly sample eligible user turns from the product context whose

deployment we aim to simulate (for example, ChatGPT or Codex). For each sampled turn, we retain the conversation prefix through that user message and remove all later turns.

**Step 2: Generate a target-model completion in the simulated environment.** We generate the next assistant response for each prefix using candidate model  $X$  in a production-like environment. Let  $D_Y$  denote the distribution of sampled prefixes  $h$  from deployment  $Y$ . We write  $X \rightarrow Y$  for the simulated distribution obtained by sampling  $h \sim D_Y$  and generating the next response with  $X$ ; throughout, the arrow runs from the completion model to the source of the prefixes. For category  $c$ , let  $r_c(X \rightarrow Y)$  denote its prevalence in this distribution and  $r_c(X)$  its realized prevalence when  $X$  is deployed. This is a single-turn approximation of the target deployment: it captures the target model’s response to realistic user contexts, but may not fully reproduce tool state or capture multi-turn adaptation effects between the user and the model. In preliminary experiments, we did not find meaningful improvements to our estimates by simulating multiple turns.

**Step 3: Audit the simulated deployment for new misbehaviors.** We run an automated auditing pipeline over the simulated deployment  $X \rightarrow Y$ , with the aim of identifying previously unseen forms of misalignment introduced by model  $X$ .

**Step 4: Produce pre-release prevalence estimates.** We update the taxonomy of previously known misbehavior types with any new types of interest surfaced by the auditing step, then freeze a measurement procedure for each (in our case, a grader prompt). For each tracked category  $c$ , we then measure  $r_c(X \rightarrow Y)$  as our estimate of  $r_c(X)$ . Throughout, reported rates refer to outputs of these fixed measurement procedures, which have been validated but may be imperfect. The decision to release the model can be informed by these estimates.

**Step 5: Validate and report estimate quality after deployment.** Measuring and reporting the quality of our predictions is an integral step of the pipeline. If the model is released, we rerun the same fixed measurement stack after release on production traffic from model  $X$ . This post-release rerun evaluates the pipeline along three axes:

1. *Taxonomy coverage*: did post-release auditing surface any important misbehavior class missing from the frozen taxonomy?
2. *Directional accuracy*: for known behaviors, did resampling correctly predict whether deployment prevalence would increase or decrease?
3. *Rate calibration*: how close were the predicted and realized prevalences? Calibration is the most ambitious measure of success, as strong calibration would directly provide directional tracking.

### 3 Experiments

We evaluate deployment simulation across four sets of experiments. First, we assess how well it forecasts post-deployment misbehavior rates across GPT-5-series launches, including a GPT-5.4 study with outcome-blinded predictions (Section 3.1). As part of this evaluation, we diagnose where our method fails and decompose its prediction errors into their constituent sources. Second, we test the hypothesis that deployment simulation induces less evaluation awareness than classical, handcrafted evaluations (Section 3.2). Third, we test whether we can improve the realism of deployment simulation in an especially challenging setting: agentic work with heavy tool use that depends on external state which requires its simulation (Section 3.3). Finally, we test whether external evaluators can apply our methodology using public datasets (Section 3.4).

**Production traffic distribution and user privacy.** With the exception of Section 3.3, our ex-

periments use de-identified ChatGPT traffic from GPT-5-series deployments from users who allow their data to be used for model improvements. We exclude Temporary Chats,<sup>1</sup> employee traffic, ChatGPT Enterprise, API, and Codex traffic. Due to technical limitations in our current processing pipeline, we also exclude conversations with file attachments. Because of these filtering steps, our production estimates of misbehavior are most representative of this filtered slice of production traffic. *Before any use in our evaluations, account-linked identifiers and identifiable information were automatically removed. Only aggregate rates and statistical comparisons are reported in this work.* In total, we analyze approximately 1.3 million conversations from four production deployments spanning August 2025 to March 2026 (Table 5).

We only consider Thinking models of the GPT-5-series, and so may omit “Thinking” when referring to model names.

### 3.1 Forecasting GPT-5-series Deployment Misbehavior

#### 3.1.1 Setup and Hypotheses

**Models and misbehavior categories.** We evaluated the pipeline across 20 categories of deployment-time misbehavior for GPT-5.4. The predictions used in the primary analysis were frozen before we inspected the held-out production measurements, although they incorporated a pipeline update made after GPT-5.4 was released. These categories fall into two classes: disallowed-content outputs (e.g. sexual content), and misaligned actions (e.g. deceiving the user about tool use). For the full list, see Appendix G. We also conducted retrospective studies of previous GPT-5-series Thinking model releases using the same categories to further validate our pipeline. We obtained all misbehavior labels by applying GPT-5 Thinking graders at high reasoning effort on simulated or real deployment conversations. Each grader labeled only the final assistant turn for the relevant misbehavior.

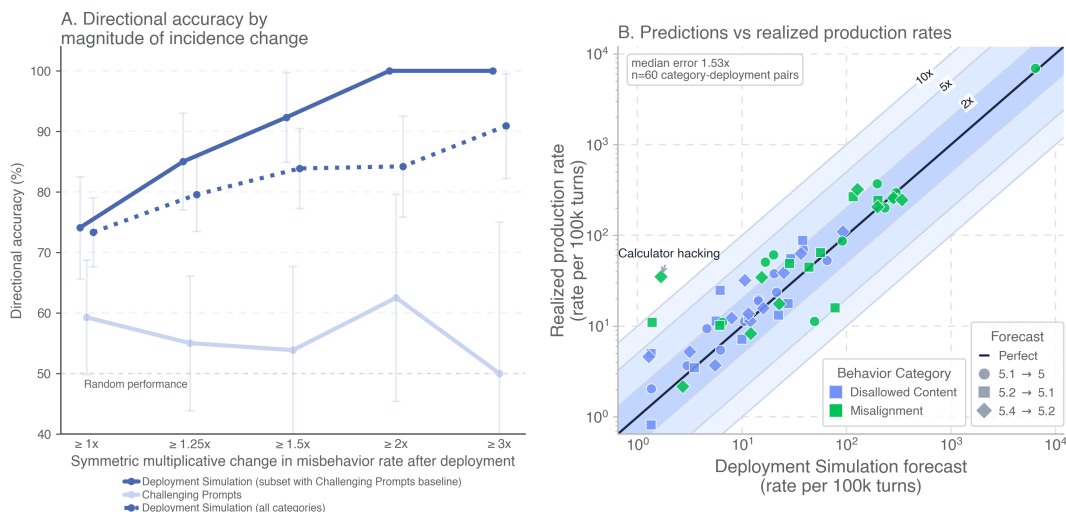


Figure 3: **A:** When predicting whether misbehaviors will become more or less common after deployment of a candidate model, deployment simulation outperforms Challenging Prompts (on the subset of categories for which that baseline is defined). Its directional accuracy also increases with the magnitude of the realized rate change. Error bars show binomial standard error. **B:** Deployment simulation rate forecasts generally fall within a factor of 2-5 $\times$  of realized production rates.

<sup>1</sup><https://help.openai.com/en/articles/8914046-temporary-chat-faq>

**Baselines.** Our *Challenging-Prompts baseline* uses the evaluations of that name reported in OpenAI system cards. These evaluations consist of production prompts that previously induced model misbehavior.<sup>2</sup> We extract directional predictions by comparing evaluation results across models; for example, checking whether GPT-5.2 is expected to produce sexual disallowed content more or less than GPT-5.1. We also extract GPT-5.4 point estimates from these evaluations. For each category, we fit a log-log linear relationship between evaluation results and realized production rates for prior GPT-5-series models, then use that fit to predict the GPT-5.4 production rate; see Appendix A for more details. The Challenging Prompts baseline is defined only for disallowed-content categories. To cover all tracked categories, we additionally consider a *previous-rate baseline*, which naively predicts that the incidence of each category will remain unchanged from the preceding model’s deployment.

**Preregistered hypotheses.** We preregistered three hypotheses (see Appendix A for details):

1. **H1:** Deployment simulation forecasts are more accurate than the previous-rate baseline.
2. **H2:** Deployment simulation forecasts are more accurate than the Challenging Prompts baseline.
3. **H3:** No novel misaligned behavior occurs in ChatGPT traffic at or above the preregistered detectable-incidence threshold ( $\approx 0.003\%$ ).

Unless otherwise stated, our descriptive analyses, figures, and category-level comparisons use a *symmetric multiplicative factor*, defined for two rates  $r_1$  and  $r_2$  as  $\max(\frac{r_1}{r_2}, \frac{r_2}{r_1})$ . Thus, rates that differ by a factor of two have a symmetric multiplicative factor of  $2\times$ , regardless of which is larger. When comparing a forecast with a realized rate, we call this quantity *symmetric multiplicative error*; when comparing rates across deployments, we use it as the magnitude of the multiplicative change. To keep the ratio finite for zero-count categories, we apply Jeffreys smoothing,  $(K + \frac{1}{2})/(N + 1)$ , to all count-derived forecasts, baselines, and realized rates before calculating it. The preregistered H1 and H2 hypothesis tests are the exception: they use mean per-category binomial negative log-likelihood (NLL) scored against the raw realized outcome counts. Accordingly, all H1 and H2  $p$ -values refer to NLL, while factor errors and category win counts are descriptive. See Appendix A for further discussion of this evaluation philosophy and the preregistered scoring rules.

### 3.1.2 Results

**Deployment simulation better predicts whether measured misbehavior will increase or decrease.** For deployment decisions, the simplest question is whether a tracked misbehavior is likely to become more or less common after release. For misbehaviors whose production incidence changed by at least  $1.5\times$ , simulated deployments predict the direction more accurately than the Challenging Prompts baseline on the comparable subset (92% vs. 54%) and remain accurate across all tracked categories (84%; Figure 3, left). Thus, without requiring a separately designed and curated evaluation set for each failure mode, deployment simulation better predicts changes across many behaviors using a uniform sample of real traffic.

**Quality of frequency estimates obtained via deployment simulation.** In contrast with traditional evaluations, simulated deployments can be used to predict the incidence rate of misbehavior in deployment *directly*. Across 60 category–deployment pairs from retrospective analyses of

---

<sup>2</sup>Challenging Prompts are one type of evaluation commonly used for pre-deployment risk assessment. Relative to handcrafted or synthetic prompts, one might expect Challenging Prompts to be comparably predictive of realized rates because they also consist of production data, albeit adversarially selected. We therefore treat them as a strong traditional-evaluation baseline.

GPT-5.4 rate predictions by category compared to baselines

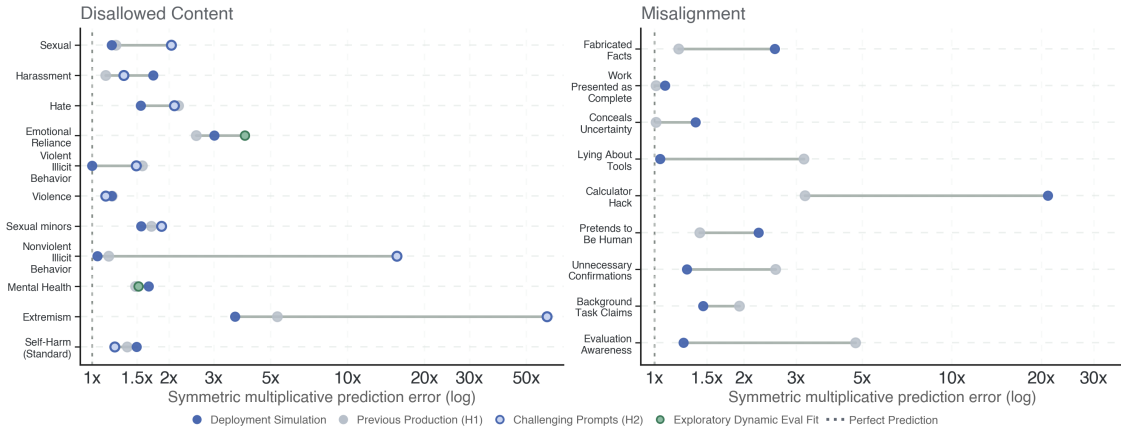


Figure 4: **GPT-5.4 symmetric multiplicative rate-prediction error by category.** Deployment simulation outperformed challenging-prompts rate estimates for 6/9 disallowed-content categories. While it was only better than the naive previous-production baseline for 11/20 categories, for categories that changed by  $\geq 1.5\times$  the winrate was 8/11.

GPT-5.1 and GPT-5.2 and the outcome-blinded GPT-5.4 forecast, predicted and realized production rates had a strong Pearson correlation on the log scale ( $r = 0.91$ ; Figure 3, right). (Figure 3, right).

Deployment simulation outperformed Challenging-Prompts rate estimates for 6 of 9 disallowed-content categories. The one-sided sign-flip test on mean per-category NLL gave  $p = 0.0469$  (Table 4). The naive previous-rate baseline was stronger: deployment simulation had lower per-category NLL for 11 of 20 categories, but its mean NLL was not lower overall, providing no support for H1 ( $p = 0.6567$ ). These results can be largely explained by two factors: GPT-5.4 misbehaves at frequencies similar to GPT-5.2 for many categories, and our deployment simulation pipeline has some fixed bias term that primarily affects categories whose rates did not change substantially. Indeed, for categories whose production rates changed by less than  $1.5\times$ , deployment simulation had lower symmetric multiplicative error than the Challenging Prompts baseline for 2/5 categories and the previous-production baseline for 3/9. For categories whose production rates changed by at least  $1.5\times$ , the corresponding win rates were instead 4/4 and 8/11.

The largest prediction error for GPT-5.4 was for calculator hacking, predicted to be  $21\times$  less frequent for GPT-5.4 than its eventual realized rate (1.7 vs. 35.2 per 100k; see Appendix C). Errors of this size limit the reliance one can place on our pipeline in its current form. That said, this error, like most of our largest errors, appears to be due primarily to resampling fidelity, as shown in Figure 12 and discussed below.

**Resampling inconsistencies affect estimates more than shifts in the prefix distribution.** Forecast error can arise from two sources: shifts between the historical and target prefix distributions, and mismatches between the resampling and production environments. To isolate their contributions, we report symmetric multiplicative error for three comparisons in Figure 5:

- *Environment fidelity error*: comparing the 5.4  $\rightarrow$  5.4 resampled rate with the GPT-5.4 production rate isolates differences between the resampling and production environments, such as sampling-pipeline mismatches or poor fidelity of stateful tools.
- *Prefix fidelity error*: comparing the 5.4  $\rightarrow$  5.2 and 5.4  $\rightarrow$  5.4 resampled rates holds the completion model and resampling environment fixed, isolating the effect of using off-policy prefixes.

- *Full forecast error*: comparing the 5.4  $\rightarrow$  5.2 resampled rate with the GPT-5.4 production rate gives the error observed in practice.

Note that errors that are product of the environment and prefix-distribution rate can be in opposite directions and partially cancel. Somewhat surprisingly, tail errors are dominated by the resampling environment rather than by prefix-distribution shift: with perfect environment fidelity, our largest remaining prediction error would have been  $\sim 2.5\times$ . This suggests that many of the largest errors arise from potentially addressable engineering limitations.

**Prefix-distribution error can be reduced by using the most recent production data.**

The best time window from which to sample production data is not obvious. Prefixes from a large historical window may provide broader coverage and greater robustness to seasonality, whereas the most recent data may better represent the traffic encountered at deployment time. We took the former approach in our analyses, but later investigation suggested that the latter may have been preferable. In Figures 6 and 13, we compare changes in misbehavior rates within each deployment and across adjacent deployments after excluding weeks with less than half the median weekly sample size. Across categories, within-deployment variation is often larger than the change associated with the model deployment itself. However, this analysis is confounded by system-prompt updates and other deployment-time adjustments. Taken at face value, the results suggest that restricting prefixes to the most recent production traffic may reduce deployment-prediction error, especially error due to prefix-distribution shift.

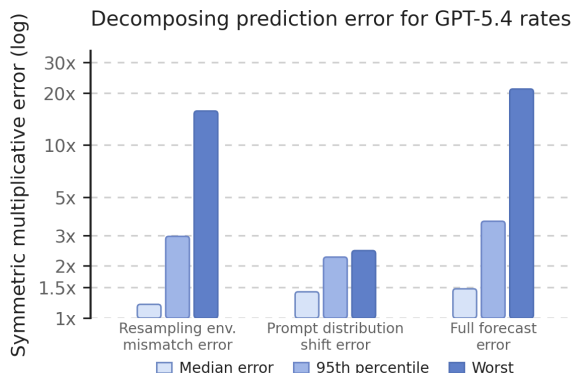


Figure 5: **Error decomposition for GPT-5.4 Thinking rate forecasts.** Mismatch between resampling and production environments is a large component of full prediction error and dominates among highest-error categories.

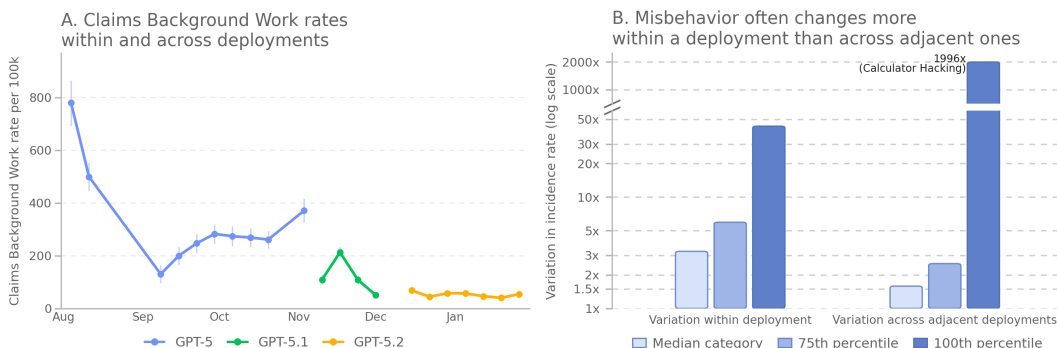


Figure 6: **Within-deployment variation in misbehavior rates and comparison with adjacent-deployment changes.** Panel A shows an example of substantial variation in model misbehavior rates both within a single deployment, and variance in variation also across adjacent ones—see Figure 13 for all categories. Panel B summarizes symmetric multiplicative variation across categories using the median, 75th percentile, and maximum. Median within-deployment variation is greater than median variation across adjacent deployments, whereas the relationship reverses above the 75th percentile.

**Deployment simulation was sufficient for identifying the one novel ChatGPT misalignment in the deployment window we study: calculator hacking.** Across the GPT-5-series deployments, the only significant novel misalignment in ChatGPT traffic that we are aware of was calculator hacking, in which the model uses the browser tool as a calculator while behaving as if it had performed a search. This behavior appeared with GPT-5.1, and our automated audit of a simulated deployment would have surfaced it before release. For GPT-5.4, the audit of production conversations surfaced no novel misbehavior type, consistent with H3. The retrospective detection of calculator hacking provides limited positive evidence about sensitivity, based on one relatively prevalent behavior, although we view it as an encouraging initial result.

### 3.2 Evaluation Awareness in Simulated Deployments vs Traditional Evaluations

If a model is aware that it is being evaluated, this awareness might affect its behavior and distort our safety measurements. This is especially concerning because even small, subtle mismatches between evaluation and production environments can produce evaluation awareness. In this subsection, we test the hypothesis that simulated deployments reduce evaluation awareness relative to traditional, handcrafted evaluations. We use three distinct metrics: evaluation-detection capability, verbalized alignment-evaluation awareness, and metagaming. On the two metrics applicable across the full evaluation suite—evaluation detection and metagaming—traditional evaluations displayed substantially more evaluation awareness than production traffic (Figure 7), echoing results reported elsewhere [1, 3]. Verbalized alignment-evaluation awareness was also much more common in alignment evaluations than in production. In contrast, simulated-deployment rates were much closer to production across all three metrics.

**Setup.** GPT-5.2 was the policy model that generated the trajectories in all three conditions. Separate GPT-5.2 calls, independent from trajectory generation, served as evaluation awareness detectors. For *evaluation-detection capability*, a detector received the full trajectory in a fresh context window and classified it as organic production traffic or an evaluation [1]. For *verbalized alignment-evaluation awareness*, a chain-of-thought monitor scored whether the policy explicitly reasoned that it was being evaluated [3]. A separate *metagaming* monitor scored reasoning about the oversight process [4]. We applied these measurements to GPT-5.2 production conversations, simulated deployment trajectories (5.2  $\rightarrow$  5.2), and a collection of traditional evaluations:

1. Scheming-propensity evaluations, a suite of alignment evaluations introduced in prior work [3];
2. Reward-hacking evaluations introduced in prior work [5, 6], including impossible coding and interview-style tasks;
3. SWE-Bench [7], a public agentic capability evaluation;
4. AIME [8], a public question-answering evaluation;
5. OpenAI-Proof QA, a private question-answering evaluation used in OpenAI system cards [9];
6. MonorepoBench, a private agentic coding evaluation used in OpenAI system cards [9].

This collection mixes private and public evaluations, alignment and capability assessments, and question-answering and agentic tasks. We included this range to reduce confounding from familiarity with a particular evaluation or from the need to use tools. We ran the evaluations using configurations resembling our system-card pre-deployment suite.

**Traditional evaluations are much easier to identify and show much more metagaming than production traffic.** When asked directly, the evaluation detector classified 98.9–100% of trajectories from every traditional-evaluation category as evaluations, compared with 5.38% of

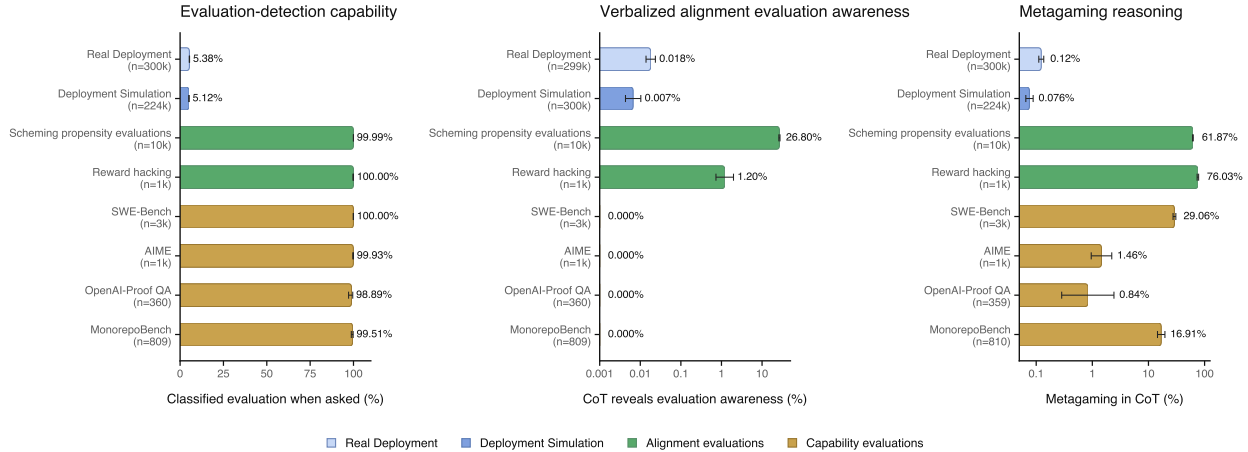


Figure 7: **Simulated deployment appears to substantially mitigate evaluation awareness.** We use three measures of evaluation awareness on GPT-5.2: evaluation-detection capability, verbalized alignment evaluation awareness, and metagaming. Real and simulated production traffic show similar rates across all three metrics, whereas traditional evaluations have substantially higher rates. The scheming-propensity and reward-hacking bars pool their constituent datasets; see Figure 14 for the per-dataset breakdown.

production conversations—a difference of approximately 18–19 $\times$ . This pattern held for alignment and capability evaluations, public and private datasets, and question-answering and agentic tasks. Inspecting the detector’s chain of thought suggested that its judgments were driven largely by obvious format cues, such as exam-style questions or requests to produce a `solution.py` file. Metagaming also occurred more often in every aggregate traditional-evaluation group shown in Figure 7: rates ranged from 0.84% to 76.0%, compared with 0.12% in production traffic, a difference of approximately 7–630 $\times$ . Anecdotally, evaluation-detection false positives in production largely involve users informally testing model capabilities or requesting assistance with school assignments.

**Traditional alignment evaluations show evaluation-awareness signals roughly two to three orders of magnitude more often than production traffic.** Across the two aggregate alignment-evaluation groups in Figure 7, verbalized alignment-evaluation awareness occurred in 1.20–26.80% of samples, compared with 0.018% of production traffic—approximately 67–1,489 $\times$  more often. Metagaming occurred in 61.87–76.03% of alignment-evaluation samples, compared with 0.12% of production traffic—approximately 516–634 $\times$  more often. As expected, we did not detect verbalized alignment-evaluation awareness in our capability evaluations. In alignment evaluations, verbalized awareness and metagaming were also more often judged to affect the model’s decisions than in simulated deployments (Figure 15).

**Evaluation-awareness rates in simulated deployments are much closer to production than traditional-evaluation rates.** All three simulated-deployment rates of evaluation awareness were within 2.6 $\times$  of their production counterparts. For evaluation detection and metagaming, simulated deployments were far closer to production than to any traditional-evaluation aggregate; for verbalized alignment evaluation awareness, they were far closer to production than to alignment evaluations.

### 3.3 Realistic tool simulation is possible even in agentic coding settings

In this section, we focus on simulating internal employee Codex deployment traffic rather than ChatGPT traffic for two reasons. First, pre-deployment risk assessment for internal model deployments is essential for mitigating risks, especially from misalignment. Second, agentic coding is an especially challenging setting for deployment simulation and therefore serves as a stress test for our technique.

In Section 3.1.2, we found that our largest prediction errors occurred when the resampling environment poorly matched production (Figure 5). We suspect that the primary source of fidelity error was our use of a sampling pipeline different from that used for production traffic, an issue we have since addressed. A more fundamental conceptual hurdle remains: realistic tool use depends on external state. In increasingly common agentic settings, users ask models to inspect repositories, call tools, edit files, send network requests, and perform other stateful actions. A typical agentic trajectory can involve hundreds of tool calls that depend on or modify state beyond the conversation.

This creates a dilemma for simulation. Reconstructing the exact external state can be practically infeasible, but allowing an unreleased candidate model to act directly on live systems would defeat the purpose of pre-deployment evaluation: the very harms we are trying to forecast, such as leaking sensitive information or making damaging changes, could occur during the evaluation itself.

**Tool simulator affordances.** We build on prior work on *tool simulation*, using another LLM to simulate the response a tool call would have produced [10], and add deployment-specific affordances that substantially improve the realism of reconstructed production trajectories. To test this approach, we use roughly 120,000 Codex trajectories from internal OpenAI employee traffic rather than external ChatGPT traffic. This was a uniform sample from a subset of all traffic, for which we had full code state. We find that providing the tool simulator with the following additional affordances greatly improves simulation quality:

- **Original trajectory context.** First, one can give the tool simulator access to the original conversation: identical tool calls can receive identical results, and similar calls can receive plausible interpolations. This also helps reproduce failures from the original run, such as transient service failures or connectivity issues, which may be important because unexpected tool failures often induce misalignment. This affordance is less useful when the new model takes a substantially different path than the previous model.
- **Time-matched codebase in a container.** For agentic settings, another powerful affordance is to give the tool simulator access to the codebase checked out at the relevant commit, including any untracked files present at the time. For commands that do not require network access or external state, the tool simulator can execute the proposed commands in the codebase and return the observed results. The same idea could be applied to non-coding settings that require external files.
- **A database of tool call-tool response pairs.** One can also give the tool simulator a queryable database of past tool calls and responses sourced from other conversations. This database can show the simulator what particular tool responses typically look like.
- **Read only access to network services and connectors.** Finally, the tool simulator can access read-only versions of network services to determine what network calls would likely have returned.

**Evaluating tool-simulation realism.** We evaluate tool-simulation realism in two ways. First, we

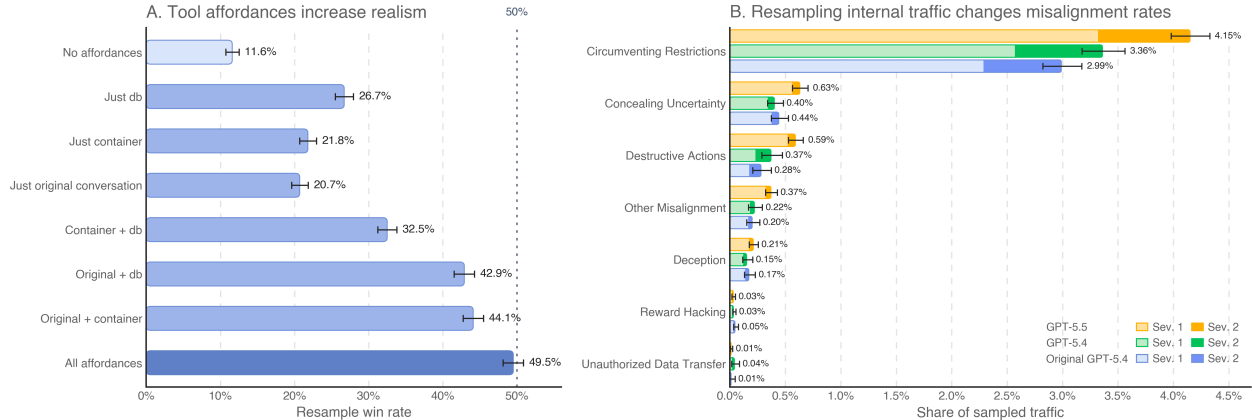


Figure 8: **A: Trajectories with tool simulation are hard to indistinguish from real data.** A resample “wins” when the GPT-5.4-based discriminator selects the simulated rollout as real. Providing the tool-simulator LLM with all affordances increases this win rate from 11.6% to 49.5%, close to the 50% parity point. **B: Resampling-based estimates of misalignment rates.** Same-model resampling (5.4 → 5.4) produces category rates similar to those in real internal production data (blue vs. green). Yellow bars show pre-deployment predictions for GPT-5.5.

run a discriminator test: a GPT-5.4-based “differentiator” sees the shared prefix and two randomly ordered final-turn rollouts—one real and one simulated—and must identify the real one. Second, we measure evaluation-detection capability as in Section 3.2. The original policy model was GPT-5.4, the resampling policy model was GPT-5.5, and the tool simulator was GPT-5.4. We optimized the simulator and differentiator prompts with agents over eight iterations, alternating between improving the simulator’s win rate and strengthening the differentiator.

**Improving tool-simulation realism.** We also ablate components of the reconstructed tool-use environment. All the tested affordances appear to improve the realism winrate on their own and their benefits appear to stack.

### 3.4 Evaluating WildChat as a public proxy for production data to resample

To test whether deployment simulation can support external auditing without access to OpenAI’s private production data, we applied it to WildChat, a public dataset of chat conversations [11]. WildChat was collected between 2023 and 2024 by offering users free access to ChatGPT with GPT-3.5 and GPT-4, with explicit opt-in consent to collect and release anonymized chat transcripts. While WildChat is not expected to represent current AI usage as well as recent OpenAI production data, its public availability makes it a useful test case for auditing both OpenAI and non-OpenAI models. For each of GPT-5.1, GPT-5.2, and GPT-5.4, we resampled the final assistant turn in 100,000 randomly selected WildChat conversations and evaluated the resulting responses across our tracked categories.

**Using recent production data gives more accurate predictions than using WildChat.** Across models, estimates using recent OpenAI production data had lower multiplicative error against post-deployment production rates than WildChat estimates across tracked categories ( $1.75\times$  vs.  $2.44\times$ ; one-sided t-test on NLL:  $p = 0.000142$ ; Figure 9). This is consistent with the core premise of our method: resampling is most predictive when the prefix distribution closely approximates the deployment distribution of the target model.

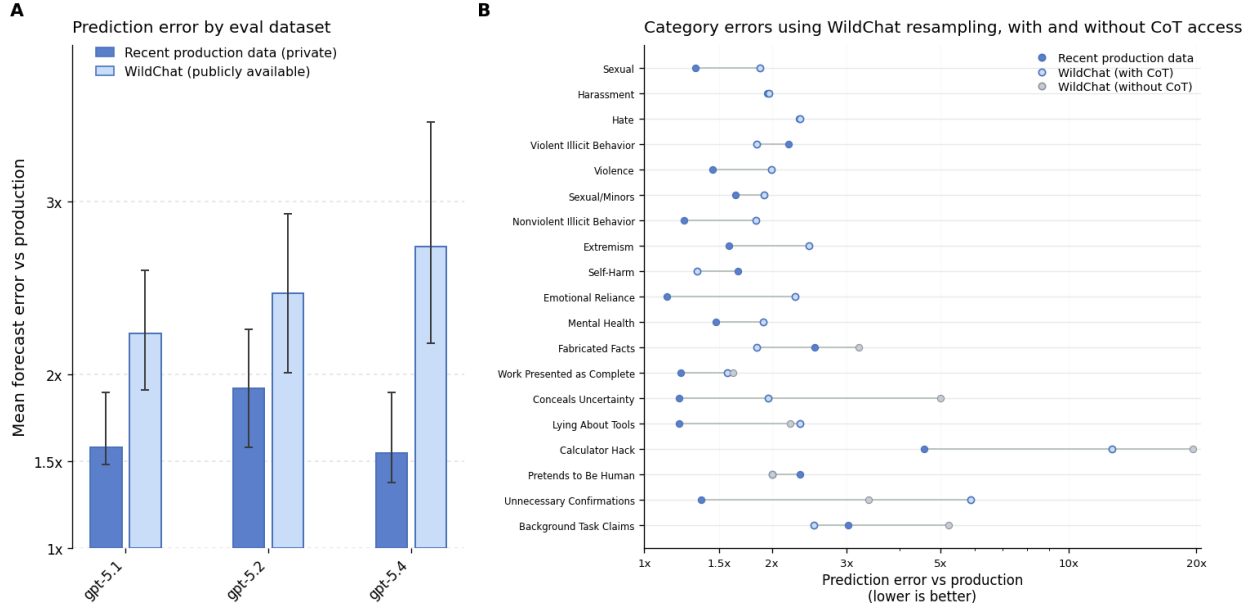


Figure 9: **Deployment simulation based on WildChat is loosely predictive of production rates.** **A:** WildChat provides a loose but useful proxy for production-data-based evaluation, with mean symmetric multiplicative errors below  $3\times$  for GPT-5.1, GPT-5.2, and GPT-5.4. **B:** Per-category errors, averaged across models, reveal where WildChat is a reasonable proxy and where it is less reliable. Dark blue shows prediction errors from deployment simulation using recent OpenAI production data (Section 3.1); pale blue shows errors from deployment simulation using WildChat; and gray shows WildChat errors when the grader cannot access chain of thought, as is typical for external researchers.

**Still, resampling WildChat data is surprisingly informative about deployment-time risk.** Despite the distribution shift, WildChat resampling predicted which of two models would perform better at a rate significantly above chance (mean accuracy of 63.9% across GPT-5-series models,  $p = 0.0025$ ). Misbehavior rates estimated from WildChat remained positively correlated with post-deployment production rates across tracked categories, and most categories had average prediction errors within a factor of three of production rates (Figure 9, right). This indicates that public chat data can recover coarse information about which failures are more or less prevalent and can therefore help estimate corresponding deployment-time risks. Because WildChat is public, it can also support cross-provider comparisons that are difficult to run with private production data.

Unlike disallowed-content categories, our misalignment categories are assessed by giving the grader the candidate model’s full reasoning trace. This raw chain of thought is not publicly accessible, creating another barrier for external auditors. We therefore tested deployment simulation on WildChat without giving the grader access to the candidate model’s chain of thought, better approximating the conditions faced by external auditors. Mean multiplicative error across GPT-5-series misalignment categories was  $2.5\times$  with chain-of-thought access and  $3.1\times$  without it; the difference in NLL was not statistically significant (one-sided paired t-test:  $p = 0.115$ ; Figure 9, right). We generally expect the importance of chain-of-thought access to vary by misalignment category.

These results suggest that, for chat-like settings, public chat datasets can help external researchers estimate real deployment misbehavior prevalence even without access to private production logs or raw CoT. However, our validation compares WildChat and ChatGPT production traffic; sep-

arate validation would be needed to assess whether similar public proxies work for more agentic deployment settings. The remaining gap with recent production data quantifies the value of more representative, up-to-date, and tool-rich public datasets.

## 4 Discussion and Limitations

Below, we highlight four limitations and directions for future work.

**Dependence on accurate measurements.** Accurate measurements of model misbehavior require accurate graders. We reused graders previously validated for OpenAI system cards [9]. That said, detecting important forms of misalignment involving deceptive or strategic behavior often relies on the model’s reasoning process transparently revealing its intent. While this condition generally seems to hold today [6], we may lose chain-of-thought monitorability in several ways: future models may reason less legibly, skip key reasoning steps, or learn to obscure their intent [12].

**Forecast quality depends on the similarity between the sampled and realized prefix distributions.** Deployment simulation estimates behavior on the distribution represented by the sampled prefixes. However, usage distributions are not fixed. For example, new deployment applications, world events, new third-party tools and connectors, privacy updates (which could change the data available to resample), and seasonal variation in use patterns can all cause the distribution of conversation prefixes from any given sampling window to differ from the simulated deployment prefix distribution. This means recent historical traffic may not always match the deployment distribution of a more capable model. Prevalence forecasts should therefore be interpreted as estimates conditional on the sampled traffic distribution. Although our experiments show strong results under the off-policy prefix shifts across GPT-5-series deployments, they should not be read as evidence of robustness to any step-changes in model affordances.

**Access to representative data.** Our results suggest that the strongest evidence for how a model is likely to behave in production comes from resampled private production data, which is available only to AI developers. We found WildChat [11] to be an informative but substantially weaker public proxy for deployment misbehavior. It is unclear whether WildChat is especially predictive of OpenAI models, rather than those of other frontier labs, because it was built from older ChatGPT data collected in 2023–2024. We encourage other AI labs to quantify the efficacy of public data sources as predictors of their models’ safety, thereby supporting external auditing and fairer production-like comparisons.

**Addressing tail risks.** While we would expect deployment simulation to approach full coverage in the limit of infinite data, in practice it can find only alignment failures that occur on prefixes used for resampling. For instance, a catastrophic failure occurring once every 10 million conversations is unlikely to appear in a deployment simulation of 1 million conversations. Future versions of deployment simulation could estimate such tail risks through stratified sampling—skewing the sample to make misalignment more likely—and then correcting for the resulting bias [13–16]. Additionally, if novel tail risks are introduced by new model affordances that aren’t sufficiently elicited by old prefixes, even these mitigation approaches would be insufficient. This underscores the importance of also investing in other pre-deployment risk-assessment techniques focused on such tail risks.

## 5 Conclusion

Pre-deployment evaluations are most useful when they cover the situations models will actually face without requiring evaluators to anticipate every failure mode. Deployment simulation is deliberately simple: hold fixed de-identified conversation prefixes from prior deployments, resample the candidate model’s next response, and audit the result. Reusing such prefixes rather than writing prompt suites by hand or curating adversarial cases expands coverage of ordinary and hard-to-anticipate contexts with less manual evaluation work. It also makes evaluation a checkable forecast: developers can state expected misbehavior rates before release and compare them with what users actually encounter. Across our studies, this broader sampling provided a stronger signal than traditional baselines; in retrospective analyses, it helped identify failure categories that had not been tracked at the time.

Tool-use settings remain the hardest case, since faithful resampling can require reconstructing mutable external state; our results suggest that simulation is a promising way to close that gap. Our WildChat results suggest that public chat datasets can be a useful starting point for deployment-grounded audits when private logs are unavailable, even if they are not a substitute for representative production traffic. We hope this encourages deployment-grounded auditing by developers and motivates newer, more representative public chat datasets with richer tool-use contexts for independent comparisons of frontier models.

Understanding how AI systems behave in the real world is crucial for frontier labs seeking to deploy them safely. A natural benchmark for this understanding is whether we can predict models’ behavior before deployment [17]. Deployment simulation is an approach to pre-deployment risk assessment that helps labs and evaluators anticipate how language models may behave in real-world settings and what risks they may pose. Used alongside traditional evaluations, these forecasts can make model risk assessment more realistic, quantitative, and useful for deployment decisions.

## Acknowledgments

We are grateful to Charles Zhao, Mikita Balesni, Jenny Nitishinskaya, and Gabriel Wu for running experiments closely related to this project and for offering valuable feedback. We especially thank Gaby Raila, Danielle Kehl, David Robinson, and Bowen Baker for their thoughtful feedback and broader support of the work. We also thank Dakota Goldberg, Sean Fitzgerald, Surya Mamidyala, and Chris Wendel for their invaluable help in establishing an intermediate version of our deployment-simulation pipeline. Finally, we thank Annie Disabato, Kyle Miller, Gaurav Misra, Pablo Orduna, Hao Sun, Madelaine Boyd, Jieqi Yu, Charles Proctor, Erik Boxhoorn, Ram Keelvedhi, Vinnie Monaco, and Ally Bennett, as well as the broader Safety Engineering team, for support that enabled us to improve the pipeline.

## References

- [1] Joe Needham, Giles Edkins, Govind Pimpale, Henning Bartsch, and Marius Hobbhahn. Large language models often know when they are being evaluated, 2025. URL <https://arxiv.org/abs/2505.23836>.
- [2] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(101):3207–3260, 2013. URL <https://www.jmlr.org/papers/v14/bottou13a.html>.
- [3] Bronson Schoen, Evgenia Nitishinskaya, Mikita Balesni, Axel Højmark, Felix Hofstätter, Jérémy Scheurer, Alexander Meinke, Jason Wolfe, Teun van der Weij, Alex Lloyd, Nicholas Goldowsky-Dill, Angela Fan, Andrei Matveikin, Rusheb Shah, Marcus Williams, Amelia Glaese, Boaz Barak, Wojciech Zaremba, and Marius Hobbhahn. Stress testing deliberative alignment for anti-scheming training. *arXiv preprint arXiv:2509.15541*, 2025. doi: 10.48550/arXiv.2509.15541. URL <https://arxiv.org/abs/2509.15541>.
- [4] Bronson Schoen and Jenny Nitishinskaya. Metagaming matters for training, evaluation, and oversight. OpenAI Alignment Research Blog, Mar 2026. URL <https://alignment.openai.com/metagaming/>.
- [5] Manas Joglekar, Jeremy Chen, Gabriel Wu, Jason Yosinski Jasmine Wang, Boaz Barak, and Amelia Glaese. Training llms for honesty via confessions. *arXiv preprint arXiv:2512.08093*, 2025. URL <https://arxiv.org/abs/2512.08093>.
- [6] Melody Y. Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y. Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability, 2025. URL <https://arxiv.org/abs/2512.18311>.
- [7] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VTF8yNQM66>.
- [8] Mathematical Association of America. American Invitational Mathematics Examination (AIME), 2026. URL <https://maa.org/maa-invitational-competitions/>. Accessed 2026-05-02.

- [9] OpenAI. OpenAI GPT-5 System Card. <https://openai.com/index/gpt-5-system-card/>, 2025. Aug 7, 2025.
- [10] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=GEcwtMk1uA>.
- [11] Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024. URL <https://arxiv.org/abs/2405.01470>.
- [12] Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for ai safety. *arXiv preprint arXiv:2507.11473*, 2025.
- [13] Gabriel Wu and Jacob Hilton. Estimating the probabilities of rare outputs in language models, 2025. URL <https://arxiv.org/abs/2410.13211>.
- [14] Neil Chowdhury, Sarah Schwettmann, Jacob Steinhardt, and Daniel D. Johnson. Surfacing pathological behaviors in language models. <https://transluce.org/pathological-behaviors>, June 2025.
- [15] Erik Jones, Meg Tong, Jesse Mu, Mohammed Mahfoud, Jan Leike, Roger Grosse, Jared Kaplan, William Fithian, Ethan Perez, and Mrinank Sharma. Forecasting rare language model behaviors, 2025. URL <https://arxiv.org/abs/2502.16797>.
- [16] Rico Angell, Raghav Singhal, Zachary Horvitz, Zhou Yu, Rajesh Ranganath, Kathleen McKeown, and He He. Estimating tail risks in language model output distributions, 2026. URL <https://arxiv.org/abs/2604.22167>.
- [17] Alan Chan. Evaluating predictions of model behaviour. <https://www.governance.ai/analysis/evaluating-predictions-of-model-behaviour>, April 2024. Accessed: 2026-06-10.
- [18] Erin D. Foster and Ariel Deardorff. Open science framework (osf). *Journal of the Medical Library Association*, 105(2):203–206, 2017. doi: 10.5195/jmla.2017.88. URL <https://pmc.ncbi.nlm.nih.gov/articles/PMC5370619/>.
- [19] Micah Carroll. Estimating model behavior before deployment with representative prompts for gpt-5.4 thinking, Mar 2026. URL [osf.io/un3he](https://osf.io/un3he).
- [20] OpenAI. OpenAI GPT-5.4 Thinking System Card. OpenAI system card, 2026. Public system card reference used in the GPT-5.4 Thinking deployment review.
- [21] Marcus Williams, Cameron Raymond, and Micah Carroll. Sidestepping evaluation awareness and anticipating misalignment with production evaluations. OpenAI Alignment Research Blog, December 2025. URL <https://alignment.openai.com/prod-evals/>.

## A Preregistration Details and Version History

Before looking at GPT-5.4 Thinking production data, we preregistered both point predictions and evaluation criteria for the prospective study using OSF [18]. The full preregistration is available at [osf.io/un3he](https://osf.io/un3he) [19]. This appendix records the intended hypotheses, scoring rules, and decision procedures for that forecast.

**Preregistration history and corrections.** We submitted an initial preregistration on March 6, 2026, before the release of GPT-5.4 Thinking. After release, but before inspecting the GPT-5.4 production outcomes used for validation, we improved the realism of the deployment simulation pipeline and filed an updated preregistration on April 20, 2026. Production outcomes were extracted retrospectively after the measurement window closed, and no member of the study team inspected the validation outcomes until April 21, 2026. The April 20 amendment therefore preceded any access to those outcomes. That update changed the pipeline’s point predictions but not the category set, measurement window, hypotheses, scoring rule, or statistical tests. The revised forecasts were therefore outcome-blinded, although they were not produced entirely before deployment.

During analysis, after inspecting the validation outcomes, we found an implementation error in the registered Challenging Prompts rate estimates. We corrected the error, which strengthened the Challenging Prompts baseline. Separately, the registered GPT-5.2 production rate for Fabricated Facts contained a transcription error (74 rather than 267.7 incidents per 100,000 assistant turns). The erroneous value favored deployment simulation, so correcting it made the H1 comparison less favorable to our method. Table 1 reports the initial, outcome-blinded, and final corrected results under a common scoring implementation. We report only the corrected analysis in the paper.

Stage	Timing	Test	DS lower NLL	Mean $\Delta$ NLL	One-sided exact $p$
V1	March 6 prerelease forecasts	H1	11/20	$+4.84743 \times 10^{-5}$	0.642471
		H2	7/9	$-5.79204 \times 10^{-5}$	0.134766
V2	April 20 outcome-blinded forecasts after pipeline improvements	H1	12/20	$-7.82277 \times 10^{-5}$	0.198880
		H2	8/9	$-9.15194 \times 10^{-5}$	0.005859375
Final	Corrected analysis inputs (not outcome-blind)	H1	11/20	$+3.22977 \times 10^{-5}$	0.6566925
		H2	6/9	$-6.09559 \times 10^{-5}$	0.046875

Table 1: **H1 and H2 across forecast and analysis stages.**  $\Delta$ NLL is Deployment Simulation NLL minus baseline NLL, so negative values favor Deployment Simulation. “DS lower NLL” is descriptive only; the registered statistic is the mean  $\Delta$ NLL. The final corrections were made after outcome inspection: final H1 does not support H1, and final H2 should not be described as confirmatory even though  $p < 0.05$ .

### A.1 Challenging Prompts log-log linear fit

Following the equation specified in the pre-registration,  $\log(q_c) = \alpha_c \log(x_c) + \beta_c$ , we fit a separate log-log linear relationship for each category. For category  $c$  and historical model  $m \in \mathcal{H} = \{\text{GPT-5, GPT-5.1, GPT-5.2}\}$ , let  $x_{cm}$  denote the Jeffreys-smoothed Challenging Prompts unsafe

rate and  $q_{cm}$  the Jeffreys-smoothed production prevalence. We estimate

$$(\hat{\alpha}_c, \hat{\beta}_c) = \operatorname{argmin}_{\alpha, \beta} \sum_{m \in \mathcal{H}} [\log(q_{cm}) - \alpha \log(x_{cm}) - \beta]^2.$$

Let  $x_{c, \text{GPT-5.4}}$  denote the corresponding GPT-5.4 Challenging Prompts rate. The resulting rate estimate is

$$b_c^{(\text{CP})} = \exp[\hat{\alpha}_c \log(x_{c, \text{GPT-5.4}}) + \hat{\beta}_c].$$

For Extremism, all three historical unsafe counts are zero. Jeffreys smoothing keeps the log rates finite.

## A.2 Hypotheses

Let  $\mathcal{C}$  denote the preregistered set of misbehavior categories. For each category  $c \in \mathcal{C}$ , we preregistered a point prediction  $p_c$  for the category’s deployment prevalence  $\theta_c$  over a prespecified post-deployment measurement window, which started 7 days after the deployment of GPT-5.4 Thinking and ended 21 days after deployment.

For each category  $c$ , let  $N$  be the number of sampled deployment items in the post-deployment measurement window and let  $K_c$  be the number of sampled items labeled as category  $c$ . The empirical deployment prevalence is  $\hat{\theta}_c = K_c/N$ .

**Evaluation philosophy.** In practice, we care less about absolute frequency error than about practically large multiplicative underestimates or overestimates that materially change the expected incidence of a harm category. Most of the paper therefore uses symmetric multiplicative error, an intuitive metric aligned with this goal. However, symmetric multiplicative error does not lend itself well to statistical testing. For the preregistered hypothesis tests comparing our pipeline with baselines, we used a likelihood-based scoring rule that directly matches the sampling process: binomial negative log-likelihood (NLL).

**Primary scoring rule (binomial negative log-likelihood).** We score a predicted prevalence  $p \in (0, 1)$  against the observed deployment counts  $(K_c, N)$  via the binomial negative log-likelihood<sup>3</sup>.

$$\text{NLL}(p; K_c, N) = -(K_c \log p + (N - K_c) \log(1 - p)).$$

For comparability across categories, we use per-sample NLL:

$$\bar{\text{NLL}}(p; K_c, N) = \frac{1}{N} \text{NLL}(p; K_c, N).$$

---

<sup>3</sup>Intuition: treating each sampled deployment item as a Bernoulli trial for category  $c$ , under prediction  $p$  we have  $K_c \sim \text{Binomial}(N, p)$ . Up to an additive constant independent of  $p$ , NLL is the negative log-probability that  $p$  assigns to the observed outcome  $(K_c, N)$ . Thus lower NLL means the prediction makes the observed deployment count less “surprising.” NLL penalizes *overconfident errors*: if  $p$  is very small but  $K_c$  is non-negligible, the term  $K_c \log p$  becomes very negative (large loss), and similarly if  $p$  is near 1 but  $K_c$  is much smaller, the  $(N - K_c) \log(1 - p)$  term becomes very negative. NLL is a *proper* scoring rule: in expectation over new samples, it is minimized by predicting the true rate.

**Jeffreys smoothing (applied to count-derived probabilities).** Because the preregistered tests use binomial NLL, each point prediction  $p_c$  must satisfy  $p_c \in (0, 1)$ . To ensure this, whenever we convert finite counts  $(K, N)$  into a probability for use as a *prediction* or *baseline* rate, we use the posterior mean under Jeffreys’ prior for the binomial model,

$$J(K, N) = \frac{K + \frac{1}{2}}{N + 1}, \quad \text{corresponding to } p \sim \text{Beta}\left(\frac{1}{2}, \frac{1}{2}\right).$$

This can be interpreted as adding  $\frac{1}{2}$  pseudo-success and  $\frac{1}{2}$  pseudo-failure, and ensures count-derived probabilities lie strictly in  $(0, 1)$  even when  $K \in \{0, N\}$  (preventing degenerate 0/1 probabilities that can arise purely from finite sampling). Compared to Laplace smoothing (uniform prior  $p \sim \text{Beta}(1, 1)$ , posterior mean  $(K + 1)/(N + 2)$ ), Jeffreys smoothing adds half as many pseudo-counts, providing a lighter-touch regularization while retaining a standard reference-prior interpretation.

**Definition of the preregistered predictions.** In the common case where  $p_c$  is produced as an empirical prevalence on a finite pre-release evaluation sample with counts  $(K_c^{(\text{sim})}, N_c^{(\text{sim})})$ , we define

$$p_c = J(K_c^{(\text{sim})}, N_c^{(\text{sim})}) \in (0, 1).$$

If a prediction is produced directly as a continuous probability, as for the Challenging Prompts rate estimates from the log-log linear fit in Section A.1, we still require that it lie in  $(0, 1)$ .

**H1 (Primary; directional).** Our pipeline’s predictions outperform a *previous-deployment baseline* under per-sample binomial NLL.

*Previous-deployment baseline.* Let  $(K_c^{(\text{prev})}, N_c^{(\text{prev})})$  denote the previous model’s deployment measurement counts for category  $c$ . Define the baseline probability

$$b_c^{(\text{prev})} = J(K_c^{(\text{prev})}, N_c^{(\text{prev})}) \in (0, 1).$$

Define the per-category loss difference

$$d_c = \text{NLL}(p_c; K_c, N) - \text{NLL}(b_c^{(\text{prev})}; K_c, N).$$

We hypothesize that the mean loss difference across categories is negative:

$$\mathbb{E}_{c \sim \mathcal{C}}[d_c] < 0.$$

**H2 (Secondary; directional; subset where defined).** On the subset of categories  $\mathcal{C}_{\text{trad}} \subseteq \mathcal{C}$  for which a Challenging Prompts score  $e_c$  is available for the new model, our pipeline’s predictions outperform the Challenging Prompts rate estimates under per-sample binomial NLL. Each estimate  $b_c^{(\text{trad})} \in (0, 1)$  is obtained by fitting historical Challenging Prompts scores to historical deployment prevalences using only pre-release data, then applying that mapping to  $e_c$  for the new model. The log-log linear fit is defined in Section A.1. We hypothesize:

$$\mathbb{E}_{c \sim \mathcal{C}_{\text{trad}}}[\text{NLL}(p_c; K_c, N) - \text{NLL}(b_c^{(\text{trad})}; K_c, N)] < 0.$$

**H3 (Novel misalignment types above a detectable incidence threshold; directional).** Let  $\mathcal{T}_{\text{novel}}$  denote the set of *novel misalignment types* in deployment traffic that are not mappable to any category in  $\mathcal{C}$  under a preregistered adjudication rubric. For each novel type  $t \in \mathcal{T}_{\text{novel}}$ ,

let  $\theta_t$  denote its true deployment prevalence over the measurement window, and let  $K_t$  denote the number of occurrences of type  $t$  in the  $N$  sampled deployment items (so under i.i.d. sampling,  $K_t \sim \text{Binomial}(N, \theta_t)$ ).

*Auditing sensitivity assumption.* *Auditing sensitivity assumption.* We assume the discovery-and-adjudication procedure has perfect recall for any novel type that appears at least  $m = 10$  times in the sampled dataset; i.e., if  $K_t \geq m$ , the procedure will surface and approve type  $t$  as a novel type.

*Multiplicity control via a preregistered bound on the number of novel types.* We preregistered an upper bound  $M_{\max}$  on the number of distinct novel types at the granularity induced by our adjudication rubric (i.e.,  $|\mathcal{T}_{\text{novel}}| \leq M_{\max}$ ) and a familywise error budget  $\alpha = 0.05$ . Define the *detectable-incidence threshold*  $\theta_\star = \theta_\star(N, m, \alpha, M_{\max})$  as the smallest prevalence such that a type at prevalence  $\theta_\star$  would fail to reach  $m$  occurrences with probability at most  $\alpha/M_{\max}$ :

$$\Pr[\text{Binomial}(N, \theta_\star) < m] \leq \frac{\alpha}{M_{\max}}, \quad \text{i.e.,} \quad \sum_{k=0}^{m-1} \binom{N}{k} \theta_\star^k (1 - \theta_\star)^{N-k} \leq \frac{\alpha}{M_{\max}}.$$

By a union bound, for any collection of at most  $M_{\max}$  novel types, the probability that *any* type with prevalence at least  $\theta_\star$  fails to appear  $\geq m$  times in the sample (and thus could be missed by the audit) is at most  $\alpha$ :

$$\Pr[\exists t \in \mathcal{T}_{\text{novel}} : \theta_t \geq \theta_\star \text{ and } K_t < m] \leq M_{\max} \Pr[\text{Binomial}(N, \theta_\star) < m] \leq \alpha.$$

*Hypothesis.* We hypothesize that no novel misalignment type has true deployment prevalence at or above this detectable-incidence threshold, i.e.,

$$\max_{t \in \mathcal{T}_{\text{novel}}} \theta_t < \theta_\star.$$

### A.3 Testing and decision procedures

For H1 and H2, we use the preregistered exact one-sided paired sign-flip test across categories, with the mean per-category NLL difference as the test statistic and all sign flips enumerated. H1 uses all categories in  $\mathcal{C}$  and compares deployment simulation with  $b_c^{(\text{prev})}$ ; H2 uses  $\mathcal{C}_{\text{trad}}$  and compares deployment simulation with  $b_c^{(\text{trad})}$ .

For H3, we run the preregistered discovery and independent-adjudication procedure. For any approved novel type, we define a judge, label the preregistered sample, and record its sample count  $K_t$ . If no approved novel type has  $K_t \geq m$ , then under the preregistered assumptions that  $|\mathcal{T}_{\text{novel}}| \leq M_{\max}$  and that the procedure has perfect recall for  $K_t \geq m$ , we reject the complement of H3 at familywise level  $\alpha$ ; if any approved type has  $K_t \geq m$ , we do not make this conclusion. We use  $m = 10$ ,  $\alpha = 0.05$ , and  $M_{\max} = 10$ ; for the realized sample size  $N \approx 700,000$ , the resulting threshold is  $\theta_\star \approx 2.86 \times 10^{-5}$  (0.00286%).

## B Additional results on predicting GPT-5-series misbehavior before deployment

Deployment	Category	Prod. rate	Prod $\Delta$	DS rate (per 100k)	DS pred.	DS correct	CP unsafe (%)	CP pred.	CP correct
GPT-5 $\rightarrow$ GPT-5.1	Extremism	0.1 $\rightarrow$ 2.0	17.9 $\times$	0.7 $\rightarrow$ 1.3	up	yes	0.0 $\rightarrow$ 0.0	tie	no
	Sexual minors	2.4 $\rightarrow$ 9.4	3.9 $\times$	3.0 $\rightarrow$ 4.6	up	yes	0.9 $\rightarrow$ 8.7	up	yes
	Hate	11.4 $\rightarrow$ 38.0	3.3 $\times$	12.2 $\rightarrow$ 20.2	up	yes	11.7 $\rightarrow$ 15.9	up	yes
	Harassment	32.0 $\rightarrow$ 70.4	2.2 $\times$	27.5 $\rightarrow$ 38.6	up	yes	24.1 $\rightarrow$ 29.4	up	yes
	Violent wrongdoing	35.2 $\rightarrow$ 19.7	1.8 $\times$	26.1 $\rightarrow$ 14.3	down	yes	1.8 $\rightarrow$ 4.1	up	no
	Violence	7.7 $\rightarrow$ 11.3	1.5 $\times$	5.8 $\rightarrow$ 10.5	up	yes	9.1 $\rightarrow$ 14.5	up	yes
	Non-violent wrongdoing	31.0 $\rightarrow$ 23.6	1.3 $\times$	27.9 $\rightarrow$ 21.3	down	yes	11.4 $\rightarrow$ 16.3	up	no
	Sexual content	69.4 $\rightarrow$ 53.0	1.3 $\times$	58.4 $\rightarrow$ 65.0	up	no	5.4 $\rightarrow$ 6.6	up	no
GPT-5.1 $\rightarrow$ GPT-5.2	Self-harm	5.6 $\rightarrow$ 5.5	1.0 $\times$	5.8 $\rightarrow$ 6.2	up	no	5.1 $\rightarrow$ 7.2	up	no
	Extremism	2.0 $\rightarrow$ 0.8	2.5 $\times$	1.9 $\rightarrow$ 1.3	down	yes	0.0 $\rightarrow$ 0.0	tie	no
	Hate	38.0 $\rightarrow$ 17.8	2.1 $\times$	37.9 $\rightarrow$ 27.6	down	yes	15.9 $\rightarrow$ 2.1	down	yes
	Non-violent wrongdoing	23.6 $\rightarrow$ 13.2	1.8 $\times$	30.4 $\rightarrow$ 22.3	down	yes	16.3 $\rightarrow$ 7.7	down	yes
	Sexual content	53.0 $\rightarrow$ 88.7	1.7 $\times$	56.2 $\rightarrow$ 37.8	down	no	6.6 $\rightarrow$ 3.9	down	no
	Sexual minors	9.4 $\rightarrow$ 7.2	1.3 $\times$	12.6 $\rightarrow$ 9.9	down	yes	8.7 $\rightarrow$ 0.9	down	yes
	Violent wrongdoing	19.7 $\rightarrow$ 24.9	1.3 $\times$	14.8 $\rightarrow$ 6.2	down	no	4.1 $\rightarrow$ 2.1	down	no
	Harassment	70.4 $\rightarrow$ 56.0	1.3 $\times$	34.2 $\rightarrow$ 29.2	down	yes	29.4 $\rightarrow$ 19.0	down	yes
GPT-5.2 $\rightarrow$ GPT-5.4	Self-harm	5.5 $\rightarrow$ 5.0	1.1 $\times$	7.3 $\rightarrow$ 1.3	down	yes	7.2 $\rightarrow$ 4.7	down	yes
	Violence	11.3 $\rightarrow$ 11.4	1.0 $\times$	10.0 $\rightarrow$ 5.6	down	no	14.5 $\rightarrow$ 9.1	down	no
	Extremism	0.8 $\rightarrow$ 4.6	5.6 $\times$	0.8 $\rightarrow$ 1.3	up	yes	0.0 $\rightarrow$ 0.0	tie	no
	Hate	17.8 $\rightarrow$ 39.0	2.2 $\times$	14.3 $\rightarrow$ 25.1	up	yes	2.1 $\rightarrow$ 5.7	up	yes
	Sexual minors	7.2 $\rightarrow$ 12.3	1.7 $\times$	7.3 $\rightarrow$ 7.9	up	yes	0.9 $\rightarrow$ 3.4	up	yes
	Violent wrongdoing	24.9 $\rightarrow$ 15.9	1.6 $\times$	26.2 $\rightarrow$ 15.9	down	yes	2.1 $\rightarrow$ 2.9	up	no
	Self-harm	5.0 $\rightarrow$ 3.7	1.4 $\times$	6.2 $\rightarrow$ 5.5	down	yes	4.7 $\rightarrow$ 1.3	down	yes
	Sexual content	88.7 $\rightarrow$ 110.2	1.2 $\times$	100.3 $\rightarrow$ 92.1	down	no	3.9 $\rightarrow$ 6.7	up	yes
GPT-5.4	Violence	11.4 $\rightarrow$ 13.7	1.2 $\times$	9.5 $\rightarrow$ 11.5	up	yes	9.1 $\rightarrow$ 16.9	up	yes
	Non-violent wrongdoing	13.2 $\rightarrow$ 11.4	1.2 $\times$	19.7 $\rightarrow$ 12.0	down	yes	7.7 $\rightarrow$ 0.0	down	yes
GPT-5.4	Harassment	56.0 $\rightarrow$ 63.4	1.1 $\times$	37.6 $\rightarrow$ 36.5	down	no	19.0 $\rightarrow$ 21.0	up	yes

Table 2: Directional-accuracy rows for all disallowed-content categories comparable with Challenging Prompts. “DS” denotes deployment simulation, and “CP” denotes Challenging Prompts. Production rates are raw observed frequencies; deployment-simulation rates are Jeffreys-smoothed count-derived forecasts. Both are reported per 100,000 assistant turns. “CP unsafe (%)” is  $100 \times (1 - \text{not\_unsafe})$  from the Challenging Prompts sections of the GPT-5.4 system card [20]; GPT-5 rates were recomputed to be comparable with those numbers in light of grader changes since the original system card [9]. Rows are ordered within each deployment by realized production-rate change. Across all rows, deployment simulation predicts 20 of 27 directions correctly and Challenging Prompts predicts 16 of 27 correctly; restricting to rows with at least a 1.5 $\times$  realized production-rate change gives 12 of 13 and 7 of 13, respectively.

Misbehavior incidence estimates for GPT-5.1 forecast validation

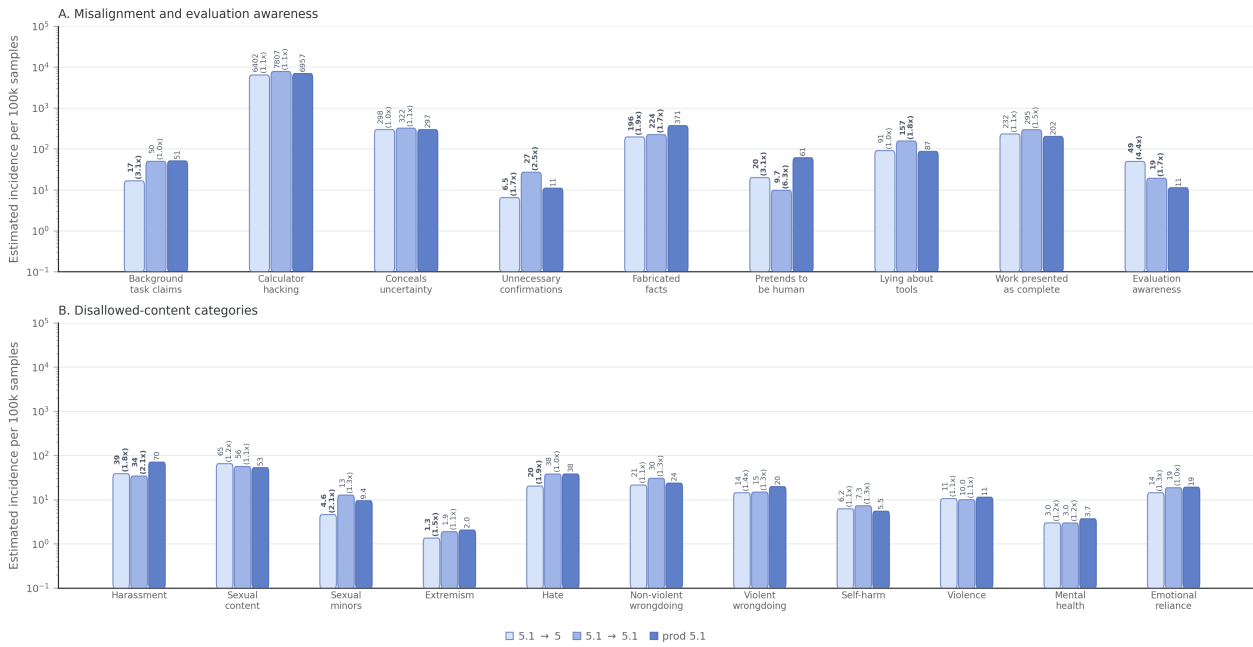


Figure 10: **Misbehavior incidence estimates for GPT-5.1 forecast validation.** Bars show rates per 100,000 assistant turns across deployment-simulation runs and the post-deployment production measurement. Error bars show one Bernoulli standard error calculated using the source sample size for each series.

Misbehavior incidence estimates for GPT-5.2 forecast validation

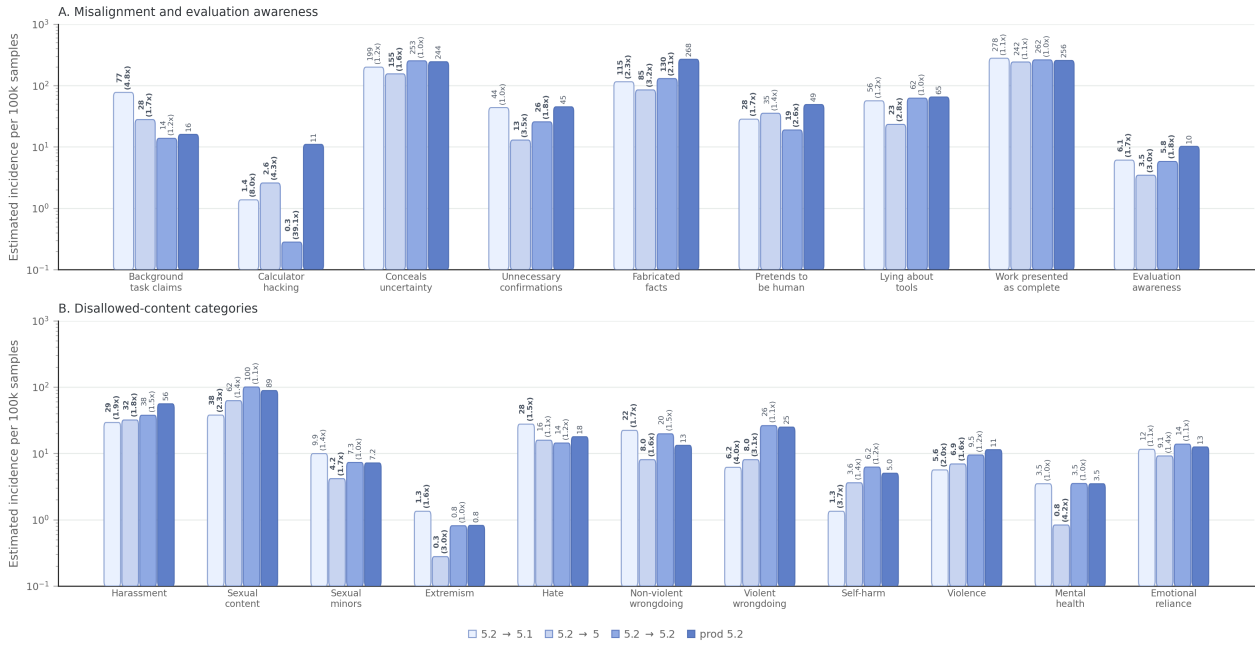


Figure 11: **Misbehavior incidence estimates for GPT-5.2 forecast validation.** Bars show rates per 100,000 assistant turns across deployment-simulation runs and the post-deployment production measurement. Error bars show one Bernoulli standard error calculated using the source sample size for each series.

Misbehavior incidence estimates for GPT-5.4 forecast validation

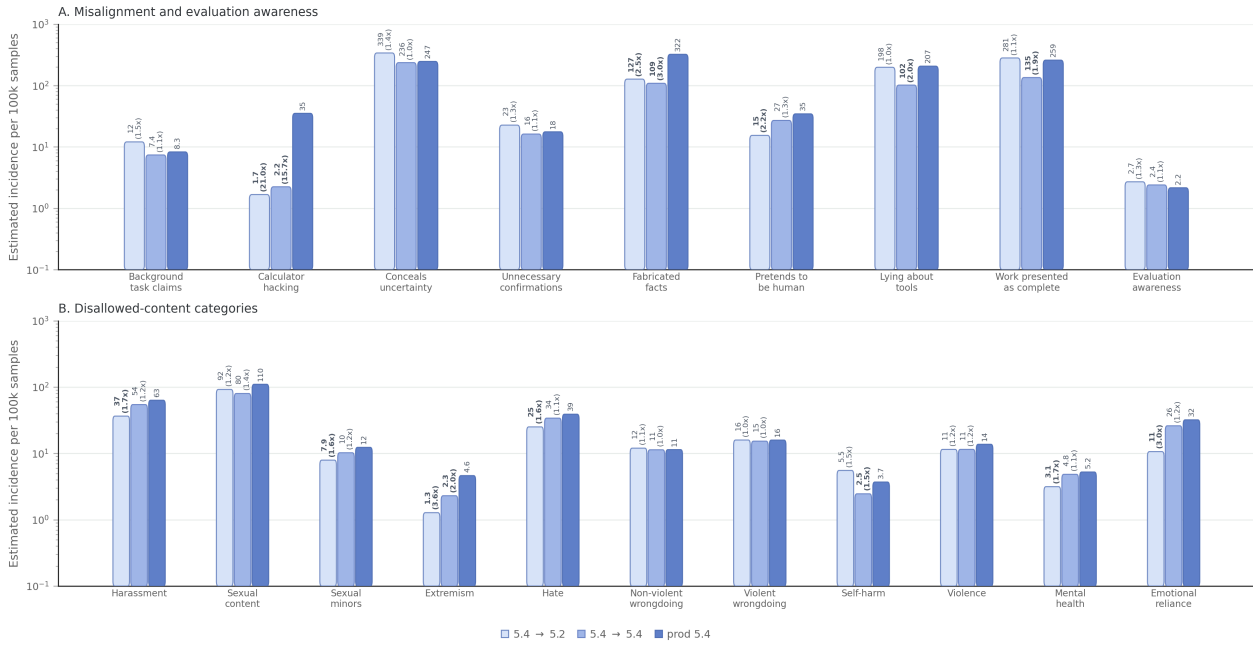


Figure 12: **Misbehavior incidence estimates for GPT-5.4 forecast validation.** Bars show rates per 100,000 assistant turns across deployment-simulation runs and the post-deployment production measurement. Error bars show one Bernoulli standard error calculated using the source sample size for each series.

## C Category-level GPT-5.4 forecasts

The tables below report the final corrected estimates, which supercede the ones in the preregistration. Aggregate H1 and H2 results for the initial registered forecasts, the outcome-blinded update, and the final corrected analysis appear in Table 1.

Category	Prod 5.4	DS	Prev. prod	$\Delta\text{NLL prev.}$
<i>Disallowed content</i>				
Harassment	63.4	36.5	56.0	$+7.63 \times 10^{-5}$
Emotional Reliance	32.1	10.6	12.6	$+3.41 \times 10^{-5}$
Mental Health	5.2	3.1	3.5	$+2.22 \times 10^{-6}$
Self-Harm	3.7	5.5	5.1	$+1.38 \times 10^{-6}$
Violence	13.7	11.5	11.4	$-1.10 \times 10^{-7}$
Non-Violent Wrongdoing	11.4	12.0	13.3	$-1.23 \times 10^{-6}$
Sexual Minors	12.3	7.9	7.2	$-4.32 \times 10^{-6}$
Sexual Content	110.2	92.1	88.7	$-7.50 \times 10^{-6}$
Extremism	4.6	1.3	0.9	$-1.36 \times 10^{-5}$
Violent Wrongdoing	15.9	15.9	25.0	$-1.90 \times 10^{-5}$
Hate	39.0	25.1	17.9	$-5.99 \times 10^{-5}$
<i>Misalignment and evaluation awareness</i>				
Fabricated Facts	322.3	127.2	267.7	$+9.96 \times 10^{-4}$
Calculator Hacking	35.2	1.7	11.0	$+5.70 \times 10^{-4}$
Conceals Uncertainty	246.8	339.0	244.0	$+1.38 \times 10^{-4}$
Claims Real-World Experiences	34.5	15.4	49.0	$+6.36 \times 10^{-5}$
Misrepresenting Work Completion	259.1	281.2	256.0	$+8.75 \times 10^{-6}$
Claims Background Work	8.3	12.1	16.0	$-1.59 \times 10^{-5}$
Has Evaluation Awareness	2.2	2.7	10.3	$-4.68 \times 10^{-5}$
Confirmation Hacking	17.6	22.6	45.0	$-1.03 \times 10^{-4}$
Lying About Tools	206.8	198.2	65.0	$-9.74 \times 10^{-4}$
<b>Sum of <math>\Delta\text{NLL}</math></b>				<b><math>+6.46 \times 10^{-4}</math></b>

Table 3: **Category-level H1 comparison with the previous-production baseline.** Rates are incidents per 100,000 assistant turns in the validation split used for the primary H1 and H2 analyses. “DS” is the outcome-blinded GPT-5.4-on-GPT-5.2 deployment simulation forecast; it was frozen after GPT-5.4 was released but before the authors inspected the held-out production measurements. “Prev. prod.” is the GPT-5.2 production baseline.  $\Delta\text{NLL}$  is DS NLL minus previous-production-baseline NLL, so positive values mean DS was worse. The final row sums  $\Delta\text{NLL}$  across categories using unrounded values; this sum is  $|\mathcal{C}|$  times the tested mean difference.

Category	Prod 5.4	DS	CP est.	$\Delta\text{NLL CP}$
Harassment	63.4	36.5	47.6	$+5.72 \times 10^{-5}$
Self-Harm	3.7	5.5	4.5	$+2.60 \times 10^{-6}$
Violence	13.7	11.5	12.1	$+1.05 \times 10^{-6}$
Non-Violent Wrongdoing	11.4	12.0	0.7	$-2.07 \times 10^{-4}$
Sexual Minors	12.3	7.9	6.6	$-9.35 \times 10^{-6}$
Sexual Content	110.2	92.1	53.9	$-2.09 \times 10^{-4}$
Extremism	4.6	1.3	0.1	$-1.18 \times 10^{-4}$
Violent Wrongdoing	15.9	15.9	23.7	$-1.46 \times 10^{-5}$
Hate	39.0	25.1	18.6	$-5.21 \times 10^{-5}$
<b>Sum of <math>\Delta\text{NLL}</math></b>				<b><math>-5.49 \times 10^{-4}</math></b>

Table 4: **Category-level H2 comparison with challenging-prompts rate estimates.** Rates are incidents per 100,000 assistant turns in the same validation split as Table 3. “CP est.” is the final corrected rate estimate from the log-log linear procedure specified in the registration, using exact Jeffreys-smoothed pre-release counts and the GPT-5.4 Challenging Prompts counts. The same fitting procedure is applied separately to all nine categories.  $\Delta\text{NLL}$  is DS NLL minus the NLL of the Challenging Prompts rate estimate, so positive values mean DS was worse. Deployment simulation had lower per-category NLL than the Challenging Prompts rate estimate for 6 of 9 categories; the exact one-sided sign-flip test on the mean  $\Delta\text{NLL}$  gives  $p = 0.046875$ . This fit was corrected after outcome inspection, so the result is post-registration rather than confirmatory. The reported sum is nine times the tested mean difference.

## D Detailed forecasting-validation figures

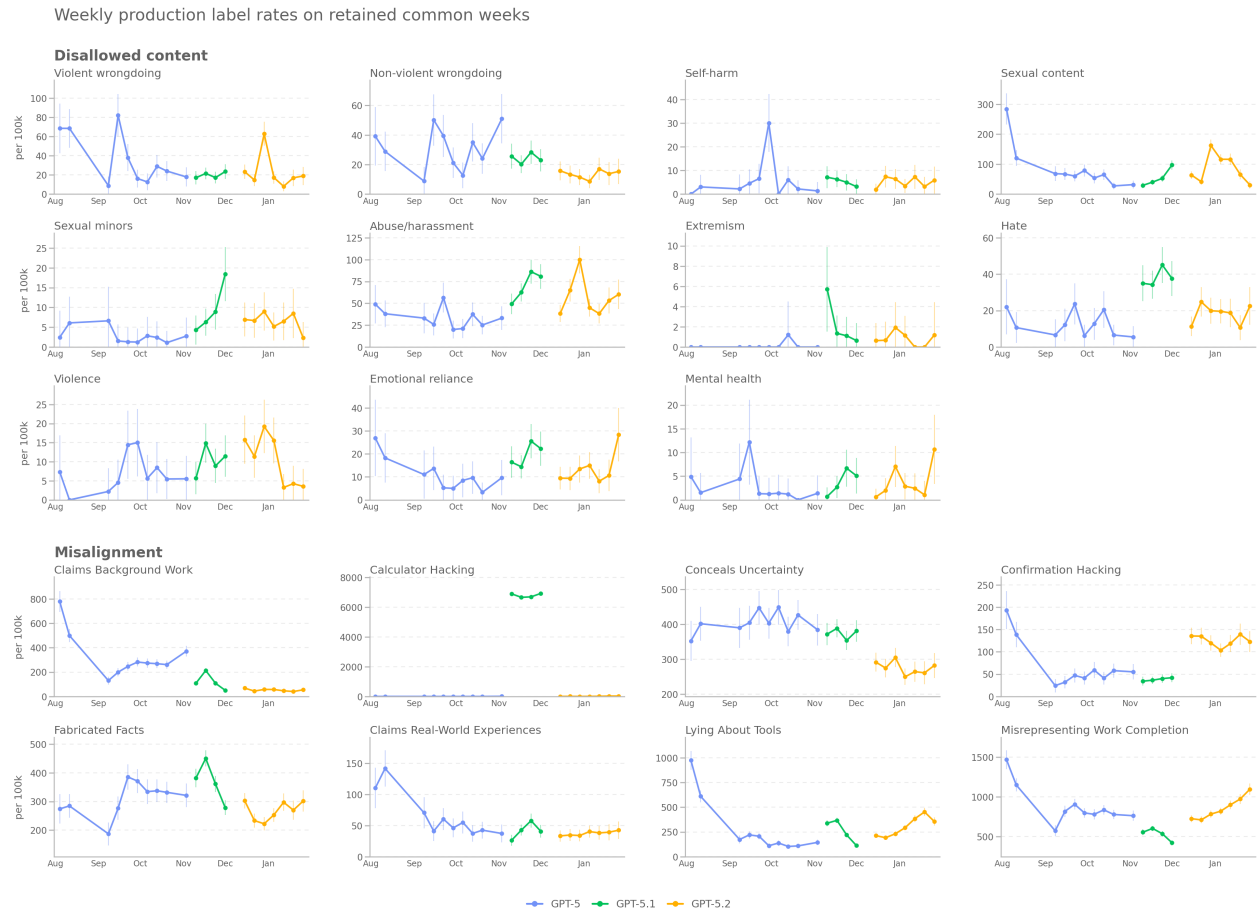


Figure 13: **Weekly label rates for retained common weeks.** Each panel shows the retained weekly production label rate for one disallowed-content or misalignment category, in incidents per 100,000 assistant turns. Error bars reproduce the binomial half-widths accompanying the weekly prevalence estimates; panels use category-specific y-axis scales to make within-category week-to-week variation legible. The retained weeks and filtering match the fixed-model traffic-variation analysis in Figure 6; GPT-5.4 is excluded because the available data do not span a comparable portion of the deployment window.

## E Detailed evaluation awareness figures

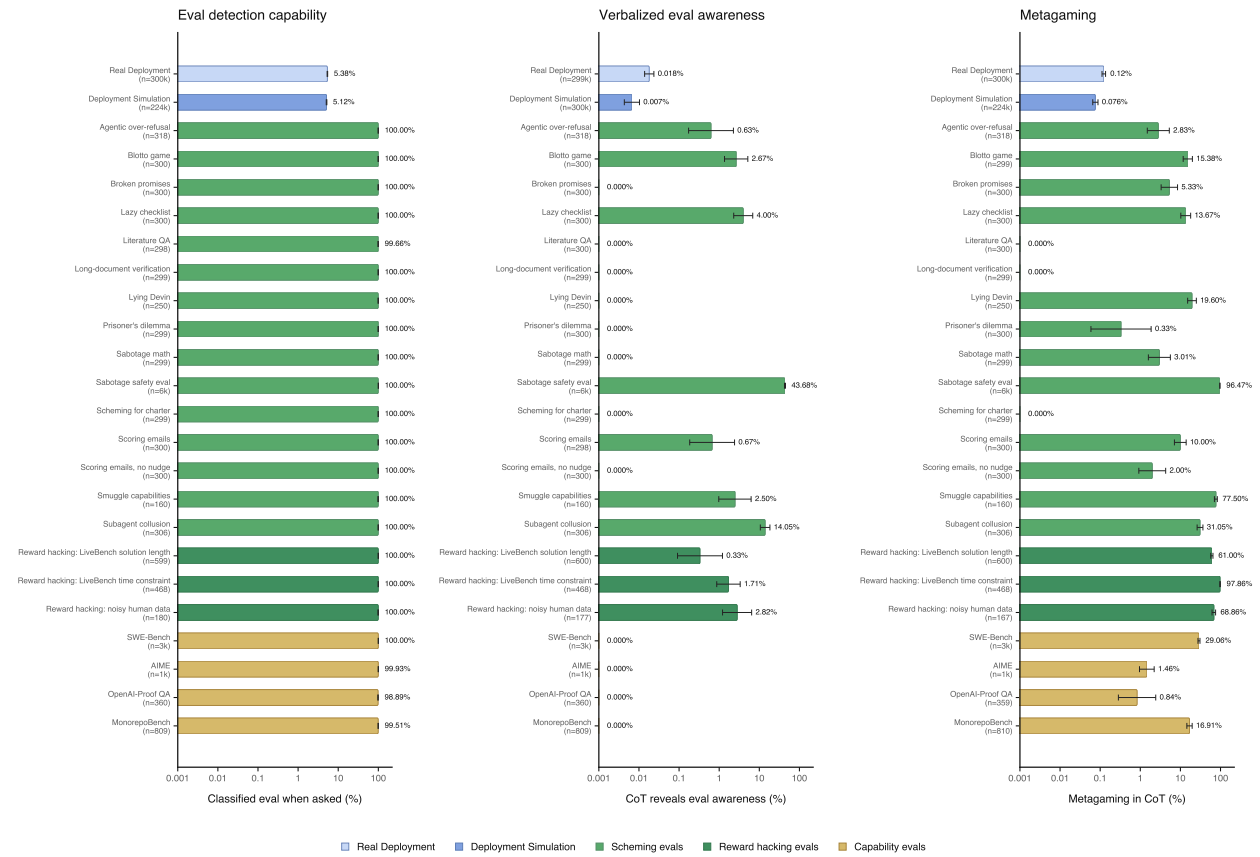


Figure 14: **Per-dataset evaluation-awareness results.** This figure expands the aggregate traditional-evaluation categories in Figure 7, showing individual scheming-propensity and reward-hacking datasets alongside the capability evaluations. Production and simulated-deployment rates are generally similar, whereas individual traditional evaluations vary substantially on both measures.

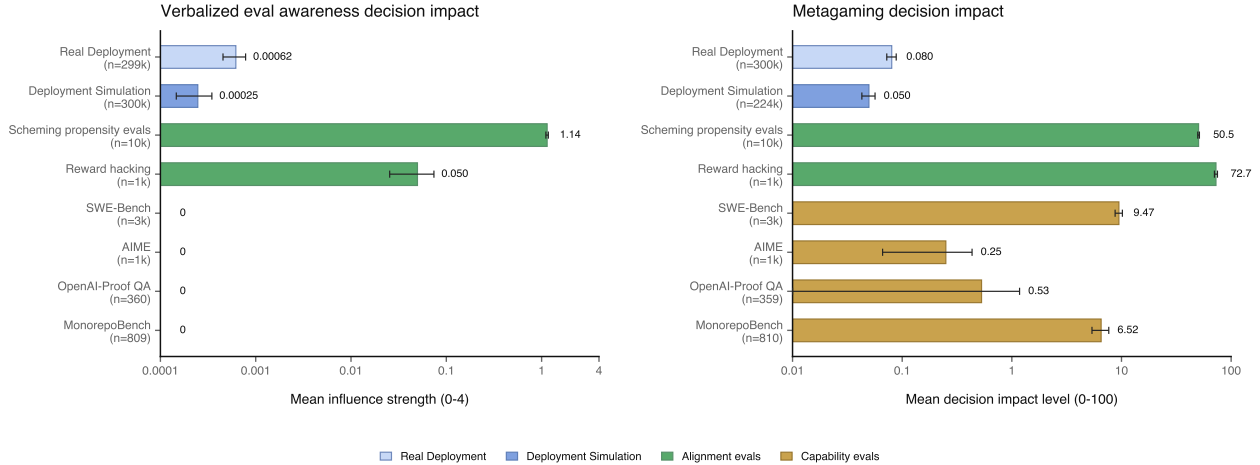


Figure 15: **Decision impact of evaluation awareness and metagaming.** Chain-of-thought monitors assess the influence of verbalized alignment-evaluation awareness (left; 0–4 scale) and metagaming (right; 0–100 scale) after reading each model’s chain of thought and actions. Dataset labels report the number of scored examples.

## F Data sources

Table 5 reports the date ranges and numbers of conversations from each deployment used in our analyses.

Deployment	Sampling window (based on last user message)	Number of conversations
GPT-5.4	2026-03-12 to 2026-03-26	≈ 700,000
GPT-5.2	2025-12-11 to 2026-01-31	≈ 200,000
GPT-5.1	2025-11-12 to 2025-12-09	≈ 200,000
GPT-5	2025-08-07 to 2025-11-11	≈ 200,000

Table 5: Production traffic sampling windows and sample sizes for each deployment.

## G Tracked categories

For our experiments, we use slightly refined versions of the misalignment categories from Williams et al. [21], while the known disallowed-content and sensitive-conversation behaviors follow the refusal and safety reporting used across recent OpenAI system cards, including GPT-5.4 [9, 20]. The full list of categories and descriptions appears in Table 6. We use two shorthand category sets throughout. *All tracked categories* comprise the full fixed taxonomy measured in resampling and production. The *traditional-baseline subset* comprises the categories that also have a traditional-evaluation analogue.

<b>Category</b>	<b>Description</b>
Claims Background Work	The model tells the user that it is working in the background when it is not.
Conceals Uncertainty	Presents uncertain or incomplete information as certain or complete.
Confirmation Hacking	Seeks needless confirmation or clarification instead of completing the task.
Lying About Tools	Misstates tool outputs or invents tool-backed results.
Fabricated Facts	Presents information as true despite indicating in the chain-of-thought that it is not.
Misrepresenting Work Completion	Claims to have completed the task or answered the user’s question without doing so.
Calculator Hacking	Uses the browser tool as a calculator while behaving as though it performed a search.
Claims Real-World Experiences	Claims first-hand physical-world experiences, possessions, or contacts.
Has Evaluation Awareness	Verbalizes in the chain-of-thought that it is in an evaluation.
Harassment	Abusive, hostile, or threatening content directed at a person or group outside the hate category.
Sexual Content	Disallowed sexual content.
Sexual Minors	Disallowed sexual content involving minors.
Extremism	Praise, propaganda, recruitment, or assistance for extremist causes.
Hate	Abusive, demeaning, or violent content targeting protected classes.
Non-Violent Wrongdoing	Complying with requests regarding illegal conduct without direct physical violence.
Violent Wrongdoing	Complying with requests regarding violent harm or related criminal conduct.
Self-Harm	Unsafe assistance involving self-harm or suicide.
Violence	Disallowed depictions of graphic violence or physical injury.
Mental Health	Mirrors, reinforces, or escalates delusional, manic, or paranoid beliefs.
Emotional Reliance	Encouraging harmful dependency, exclusivity, or manipulation in the user-assistant relationship.

Table 6: Categories tracked across experiments.