

InvokeRank: measuring how AI agents surface, select and use tools, with an instrument-validation case study on TapTax

Solomon Amos PhD
Accenture
`solomon.amos@accenture.com`

Version 1, 6 October 2026

Abstract

AI agents increasingly choose which external tool, if any, serves a request, so the host’s choice distributes software. InvokeRank is a falsifiable framework for measuring how effectively eligible tools are surfaced, selected, invoked and used to complete tasks within a defined agent environment, with the host’s own answer as an explicit alternative. It measures observable agent-distribution behaviour rather than inferring a platform’s proprietary ranking; claims about market share or user outcomes need production evidence. We formalise its nested funnel, whose product is an exact probability, and evaluate method v1.0 on TapTax, a UK tax integration owned by the author, through the Claude API and the OpenAI (ChatGPT) API, with 26,886 recorded runs. An independent reimplementations reproduces every published estimate. On the Claude and ChatGPT APIs respectively, Selection-level InvokeRank was 41.5 (90% interval 37.1 to 46.1) and 38.3 (34.5 to 42.1); 57.4% and 57.2% of positive runs used no external tool, and competitors were rarely chosen. Every selected call executed, but an uncalibrated judge counted only 57.9% and 42.6% of tasks completed. Placebo, blinded-metadata and decoy controls held; a clone control failed, because hosts chose between identical servers by name. Tool-description rewrites did not move selection; a server description, visible only on the OpenAI path, raised it from 30.5% to 72.4% on one calculator corpus and increased false invokes. Identical runs varied beyond sampling error on one host only. Native self-service judging, judge calibration, neutral eligibility and external validation remain undone, so completion, competitive and consumer-app claims are unsupported.

1 Introduction

Language-model agents now act through external tools exposed as plugins, connectors and Model Context Protocol (MCP) servers, and hosts increasingly choose among many such tools at run time: tool definitions are deferred and retrieved by a search step rather than all loaded into the prompt (Model Context Protocol, 2026; Anthropic, 2026b; OpenAI, 2026; Shi et al., 2025). For the vendor of an integration this makes the host’s choice a distribution channel. A user who asks an agent to record an expense, check a tax code or send an invoice reaches a vendor only if the host surfaces the vendor’s tool, selects it, calls it correctly and completes the task with it; otherwise the request goes to a competitor or, often, to no tool at all because the host answers from its own knowledge.

Existing evaluation does not measure this. Tool-use benchmarks test whether a model can call the right function with the right arguments given a tool set, and agent benchmarks test whether it completes multi-step tasks (Qin et al., 2024; Li et al., 2023; Patil et al., 2024, 2025; Yao et al., 2025; Luo et al., 2025; Wang et al., 2026a; Mo et al., 2026). They condition on tool use being the right answer, or score not calling as an error, and they do not ask which of

several eligible integrations a host chooses, how often it chooses none, or how stable that choice is across runs. Work on tool retrieval measures whether the right tool is retrieved (Shi et al., 2025; Qu et al., 2024; Du et al., 2024; Gan and Sun, 2025), and work on abstention and adaptive retrieval measures when a model should not consult an external source (Wen et al., 2025; Ross et al., 2025; Qian et al., 2025; Asai et al., 2024; Jeong et al., 2024); neither treats the host’s own answer as a competitor in a distribution measurement.

This paper describes and evaluates InvokeRank, a measurement method for agent-mediated tool distribution, and tests it on an integration whose owner also owns the method. Its central question is: when an agent receives a request that an integration is genuinely capable of serving, does the agent surface the integration, select it, invoke it correctly, execute it successfully and complete the user’s task, and if not, what resolved the request instead?

1.1 Research questions

- RQ1** Can agent-mediated software distribution be decomposed into a measurable nested funnel from eligibility to completion, with an explicit no-tool branch?
- RQ2** Are InvokeRank’s estimators and intervals calibrated under sparse intents, repeat clustering, cross-stage dependence and between-run variance?
- RQ3** Do placebo, blinded-metadata, clone and decoy controls detect known threats to selection validity?
- RQ4** How does selection differ across hosts and intent types, given that Surfaced is measured at different granularity on each host?
- RQ5** How often is the alternative to an external integration the host’s own answer rather than another integration?
- RQ6** Does successful tool execution predict task completion?
- RQ7** Do metadata changes cause measurable changes in selection, and at what cost in false invokes?
- RQ8** Do synthetic API measurements predict production or consumer-app behaviour?

1.2 Contributions

1. **A measurement method for agent-mediated tool distribution:** a nested funnel from Surfaced to Completed whose product is an exact probability, an explicit no-tool branch with a terminal-class split, evidence classes that are never pooled, a construction for neutral competitive eligibility, and a battery of instrument controls (Sections 3 and 4).
2. **An instrument-validation case study** on TapTax, a UK tax integration measured through the Claude API and the OpenAI (ChatGPT) API with 26,886 recorded runs over three days, in which an independent reimplementations reproduces every published estimate and the controls, replications and a simulation test where the instrument holds and where it does not (Sections 6 and 7).

1.3 Principal observations

- **The host is the main competitor.** 57.4% of positive baseline runs on the Claude API and 57.2% on the ChatGPT API used no external tool, while a competitor took the first call in 4 and 18 of 390 runs. The hosts differ in how: the Claude API often did not search (119 runs), the ChatGPT API searched and declined (186 runs). Whether those answers met the need is unknown, because the native self-service judge has not run (RQ5).

- **Selection-level InvokeRank** was 41.5 (37.1 to 46.1) on the Claude API and 38.3 (34.5 to 42.1) on the ChatGPT API, with intent scores from 1.5 to 75.9 (RQ1, RQ4).
- **Executed does not predict Completed.** Every selected call executed; the uncalibrated cross-family judge counted 57.9% and 42.6% completed, with failures in the host’s orchestration rather than the server (RQ6).
- **The clone control failed.** An exact copy of TapTax labelled `taptax-2` won none of 39 and 41 first picks: hosts decide between identical servers by name (RQ3). Placebo, blinded-metadata and decoy controls held.
- **Metadata levers are host-specific.** Tool-description rewrites never moved selection measurably; a server description naming TapTax’s calculators, which only the OpenAI path transmits, raised ChatGPT API selection from 30.5% to 72.4% and from 2.7% to 44.0% on two calculator corpora, and raised false invokes on one (RQ7).
- **Run-to-run variance is host-specific.** Identical Claude API runs varied no more than binomial sampling predicts; ChatGPT API runs across two days varied more ($p = 0.014$) (RQ2).
- **External validity is unresolved:** TapTax’s telemetry holds 0 Claude and 8 ChatGPT real tool calls in 90 days, and no human-assisted study exists (RQ8).

Section 2 reviews related work; Sections 3 and 4 formalise the method; Section 5 describes the TapTax instrument; Sections 6 and 7 report results and validity evidence; Sections 8 to 11 discuss limitations, what remains to be done, and what the evidence supports.

2 Background and related work

Tool-use and agent benchmarks. Large tool-use benchmarks test whether a model can plan and call real APIs: ToolBench gathers over sixteen thousand REST APIs and scores pass and win rates with a model evaluator (Qin et al., 2024); API-Bank separates planning, retrieving and calling (Li et al., 2023); Gorilla shows that retrieval quality bounds call correctness (Patil et al., 2024); and the Berkeley Function Calling Leaderboard scores argument validity at scale, with an abstention component that is the nearest leaderboard analogue of a negative prompt (Patil et al., 2025). Agent benchmarks score multi-step completion in stateful environments (Liu et al., 2024; Lu et al., 2025), and τ -bench judges success by the final database state and introduces pass^k , the probability that all k trials succeed (Yao et al., 2025); InvokeRank’s Reliability metric is that estimator. MCP benchmarks move evaluation onto real servers: MCP-Universe uses execution-based evaluators on eleven servers (Luo et al., 2025), MCP-Bench and MCP-AgentBench evaluate fuzzy instructions over hundreds of tools (Wang et al., 2026a; Guo et al., 2026), MCPMark reports $\text{pass}@1$ beside pass^k (Wu et al., 2026), and LiveMCPBench attributes nearly half of failures to tool retrieval (Mo et al., 2026). StableToolBench replaces unstable live APIs with a virtual server (Guo et al., 2024), the nearest precedent for our replay server. These benchmarks measure capability given a tool set and a task that needs a tool. None measures which of several eligible integrations a host chooses, nor counts the host’s own answer as a competing outcome.

Tool retrieval and deferred loading. When inventories are large, a retrieval step decides what the model sees. Strong text retrievers retrieve tools poorly (Shi et al., 2025); collaborative, reranking, hierarchical and generative retrievers address this (Qu et al., 2024; Zheng et al., 2024b; Du et al., 2024; Moon et al., 2024; Wang et al., 2025), and RAG-MCP and MCP-Zero retrieve MCP tools before or during prompting (Gan and Sun, 2025; Fei et al., 2025). Production

hosts now ship this as tool search over deferred definitions: on the Claude API a regex or BM25 search over tool names and descriptions returns tool references (Anthropic, 2026b,a), while on the OpenAI path a deferred MCP server is first visible only by its name and description (OpenAI, 2026). These documents are vendor sources; they define what Surfaced means on each host and why it is tool-level on one and server-level on the other.

Choosing not to use a tool. Abstention (Wen et al., 2025), knowledge boundaries (Kadavath et al., 2022; Yin et al., 2023; Li et al., 2025) and adaptive retrieval (Mallen et al., 2023; Jiang et al., 2023; Wang et al., 2023; Asai et al., 2024; Jeong et al., 2024) study when a model should consult an external source. For tools, MetaTool tests whether a model knows to use a tool at all before choosing one (Huang et al., 2024); When2Call scores three correct behaviours, calling, asking or declining (Ross et al., 2025); SMART shows that tools are overused on tasks a model can answer itself (Qian et al., 2025); ToolBeHonest diagnoses solvability (Zhang et al., 2024); and MCPGauge finds that models rarely use MCP tools proactively and that integration can degrade performance (Song et al., 2025). This literature optimises or scores the gate. InvokeRank instead measures how a production gate distributes requests between the host and external providers, and its native judge exists because not calling can be the right outcome.

Bias in tool and option selection. Models prefer particular option labels and positions in multiple choice (Zheng et al., 2024a; Pezeshkpour and Hruschka, 2024), and meaning-preserving format changes move accuracy widely (Sclar et al., 2024). For tools, BiasBusters measures selection among functionally equivalent tools and finds that the semantic match between query and metadata drives choice and that small description changes shift it (Blankenstein et al., 2026), and Faghih et al. (2025) show that identical tools are chosen by listing order and that assertive descriptions multiply usage. Agent shopping audits find position bias and model-version instability in the market shares agents create (Allouah et al., 2026). In information retrieval, click models separate position from relevance (Craswell et al., 2008), and fairness of exposure treats attention as an allocated resource (Singh and Joachims, 2018); Surfaced is an exposure event in this sense.

Metadata optimisation, manipulation and security. Truthful documentation improvement helps models use tools (Qu et al., 2025a), and a study of MCP tool descriptions finds most of them deficient and their repair helpful on average but not always (Hasan et al., 2026). At the other end, iterative attacks raise a tool’s selection rate several-fold (Sneh et al., 2025), MCP preference-manipulation attacks promote one server over competitors (Wang et al., 2026c), preference manipulation degrades LLM search for everyone (Nestaas et al., 2025; Kumar and Lakkaraju, 2024), and generative engine optimisation defines visibility for cited sources (Aggarwal et al., 2024). Tool metadata is also an injection channel (Greshake et al., 2023; Debenedetti et al., 2024; Wang et al., 2026b), and the MCP specification treats tool descriptions as untrusted (Model Context Protocol, 2026; Hou et al., 2026). A public distribution ranking invites optimisation, so the measurement must separate truthful optimisation from manipulation; the False-Invoke Rate and the blinded control are InvokeRank’s instruments for that.

Language models as judges. Model judges agree with human preferences at roughly the human-human rate but show position, verbosity and self-preference biases (Zheng et al., 2023; Wang et al., 2024; Panickssery et al., 2024; Ye et al., 2025), and no single judge is best across agent benchmarks (Lù et al., 2025). Prediction-powered inference corrects a judge’s rate with a small human-labelled set and gives valid intervals (Angelopoulos et al., 2023, 2026; Boyeau et al., 2025); InvokeRank’s planned completion correction is its mean estimator.

Non-determinism and evaluation variance. Hosted models are not deterministic at temperature zero (Ouyang et al., 2025; Atıl et al., 2025), partly because inference kernels are not batch-invariant (He and Thinking Machines Lab, 2025; Yuan et al., 2025). Evaluation items are a sample from a super-population, so clustered standard errors, paired differences and power analysis belong in every comparison (Miller, 2024; Madaan et al., 2024), and repeated runs of agentic evaluations differ by several points (Bjarnason et al., 2026).

Statistical foundations. The method’s statistics are standard: empirical Bayes shrinkage of many proportions (Robbins, 1956; Efron and Morris, 1975; Morris, 1983; Casella, 1985; Efron, 2010) with the beta-binomial (Skellam, 1948; Williams, 1975; Kleinman, 1973), whose plug-in intervals undercover (Kass and Steffey, 1989); Jeffreys and related binomial intervals (Jeffreys, 1946; Clopper and Pearson, 1934; Agresti and Coull, 1998; Brown et al., 2001), with design-effect deflation for clustered data (Kish, 1965; Rao and Scott, 1992; Korn and Graubard, 1998); intraclass correlation for binary data (Fleiss et al., 2003; Donner, 1986; Ridout et al., 1999); cluster bootstraps and mixed models (Davison and Hinkley, 1997; Field and Welsh, 2007; Breslow and Clayton, 1993; Bates et al., 2015; Baayen et al., 2008; Cameron et al., 2011); discrete choice and rating (Bradley and Terry, 1952; Luce, 1959; McFadden, 1974, 1978; Elo, 1978; Train, 2009; Chiang et al., 2024); and online experimentation with paired designs, variance reduction, multiplicity control and anytime-valid inference (McNemar, 1947; Connor, 1987; Deng et al., 2013; Kohavi et al., 2020; Benjamini and Hochberg, 1995; Johari et al., 2022; Howard et al., 2021; Vovk and Wang, 2021; Wang and Ramdas, 2022; Ramdas et al., 2023).

Measurement validity. Benchmarks are measurement instruments whose constructs need explicit operationalisation and validity evidence (Raji et al., 2021; Jacobs and Wallach, 2021; Bowman and Dahl, 2021; Wallach et al., 2025); contamination, saturation and Goodhart effects threaten them (Sainz et al., 2023; Golchin and Surdeanu, 2024; Ott et al., 2022; Manheim and Garrabrant, 2018); and reproducible evaluation needs pinned setups, released code and reported uncertainty (Liang et al., 2023; Biderman et al., 2024; Reuel et al., 2024). InvokeRank’s construct is “distribution by agents among eligible integrations under stated conditions”, and eligibility is its main validity risk. Staged conversion funnels (Lavidge and Steiner, 1961) and multistate models with absorbing outcomes (Andersen and Keiding, 2002; Tran-Truong and Le, 2026) are the general frame its funnel instantiates.

3 Problem formulation

3.1 Notation and the unit of measurement

An *integration* i is a software tool that an agent can call: a plugin, a connector or a Model Context Protocol (MCP) server with a list of tools. A *host surface* h is one way of running an agent: here the Claude API (Anthropic Messages API with its MCP connector) and the ChatGPT API (OpenAI Responses API with remote MCP tools). An *intent* c is a user need stated without reference to any product, for example “estimate the income tax and National Insurance on my self-employed profit”. A *prompt* p is one natural-language request written for an intent. A *run* r sends one prompt to one host with a fixed *pool* of competing integrations and records what the host did. Each prompt is run K times; we call these runs *repeats*.

The unit of the estimand is an *eligible request*: a prompt for an intent that integration i is genuinely able to serve. Section 3.4 defines eligibility, and Section 8 returns to it as the main construct-validity problem.

3.2 The nested funnel

For integration i , intent cell c and surface h , a run on an eligible prompt can reach five nested events.

Surfaced (D). The integration entered the host’s candidate set through tool search or deferred loading.

Selected (S). The host’s first external call went to the integration.

Invoked (I). The selected call carried arguments that validate against the integration’s recorded input schema.

Executed (E). The call ran on the live integration and returned a non-error, non-empty result.

Completed (C). The user’s task was completed, as judged against the intent’s stored success criteria.

The events are nested: $C \subseteq E \subseteq I \subseteq S \subseteq D$. The implementation enforces this: a run that reaches a stage without its parent is rejected as a recording error (InvokeRank, 2026a, `assertMonotone`). We write the stage rates as conditional probabilities,

$$r_D = P(D), \quad r_S = P(S \mid D), \quad r_I = P(I \mid S), \quad r_E = P(E \mid I), \quad r_C = P(C \mid E), \quad (1)$$

and the end-to-end probability as their product,

$$Q = r_D r_S r_I r_E r_C, \quad \text{InvokeRank} = 100 E[Q]. \quad (2)$$

Because the events are nested, $P(I \mid S) = P(I \mid S \cap D)$ and so on, and the chain rule gives $Q = P(D \cap S \cap I \cap E \cap C) = P(C)$. The product in Equation (2) therefore needs no independence assumption between stages. The identity also holds for the population of prompts: pooled counts estimate the marginal conditional rates, and their product is the marginal probability of the last event (Section A).

Why a product and not a weighted mean. An earlier draft of the method scored an integration by a weighted geometric mean of the five stage rates (Amos, 2026, section 15.1). Because the rates are conditional, their product is already a probability, and the weighted mean is a composite index with five defects. Its weights set arbitrary exchange rates: a 10% relative loss at one stage can count twice a 10% loss at another although both remove exactly 10% of completed tasks, so an integration with lower Q can outrank one with higher Q . Only equal weights are coherent, and then the index is $Q^{1/5}$, a monotone transform of Q with no meaning of its own. Any zero stage zeroes it, and downstream rates are undefined when an upstream stage never fires. It amplifies noise at the deep stages, whose denominators are smallest. And it cannot treat a surface that loads every tool, where $r_D = 1$ by construction, consistently with one that searches. Replacing the mean with the product is a correction and a re-application of the chain rule, not a contribution.

Most runs in this study are *selection runs*: the pool is served by a replay server that refuses every tool call, so a run stops at Selected and Invoked (Section 5). For these runs the published path reports a *Selection-level InvokeRank*,

$$\text{IR}_{DS} = 100 E[r_D r_S], \quad (3)$$

which stops at Selected and excludes Invoked by definition. In every TapTax selection run, Invoked equalled Selected (all selected calls had valid arguments), so $r_I = 1$ empirically, but IR_{DS} does not include it. We use the term Selection-level InvokeRank and the symbol IR_{DS} throughout and never call it InvokeRank without the qualifier. The method’s public thresholds forbid showing a full InvokeRank where Executed and Completed were not measured (Amos, 2026, section 15.4).

Table 1: Terminal classes of the Agent Resolution Split as implemented (InvokeRank, 2026c). Precedence for a run with no external tool: a native verdict that the need was met wins whether or not the host searched; otherwise a host that searched (or had every tool loaded) declined; otherwise the need is unresolved. Unjudged no-tool runs are excluded from the denominator and counted separately.

Class	Rule
Focal integration completed	The first external call went to the focal vendor; in execution runs it also needs Executed and a judged Completed, otherwise the run is unresolved; in selection runs the class reads “selected”
Competitor completed	The first external call went to an integration eligible for the intent (selection evidence only, since competitors are replayed)
Other external tool	The first external call went to a pool server not eligible for the intent, a control or a decoy
Native self-service completed	No external tool, and the native judge says the host’s own answer met the need
Searched then declined	No external tool; the host searched (or had every tool loaded) and the need was not met
Unresolved	No external tool, no search, need not met; or the host refused

3.3 The no-tool branch and the host as a competitor

A host can resolve a request without any external tool, for example by answering a tax question from its own parametric knowledge. Tool-use benchmarks usually condition on a tool being used, or score a refusal to call as an error (Qin et al., 2024; Patil et al., 2025; Yao et al., 2025; Ross et al., 2025); InvokeRank treats the host’s own answer as a competing outcome. Every eligible positive run receives exactly one terminal class Y from

$$Y \in \{\text{focal, competitor, other external, native self-service, searched then declined, unresolved}\}.$$

The first three classes say where the first external call went. The last three apply to runs with no external call and depend on a separate *native judge*, which decides whether the host’s own answer met the need. Table 1 gives the rules as implemented (InvokeRank, 2026c). Correct native self-service is not a success for the integration and is not counted in Q . It is reported in an *Agent Resolution Split*, with a Host Self-Serve Rate (native self-service over eligible positive runs) and a No-tool Decline Rate (searched then declined over eligible positive runs).

The native judge first classifies what the need requires: *answerable* from knowledge and the conversation, the person’s *private state*, an *action*, or *fresh data*. An answer that only explains how to do a private-state, action or fresh-data task is never counted as self-service; the rule is enforced in code after the judge returns. A natural model of the host’s choice is two-level: first whether to use any external tool, then which one,

$$P(Y = i) = P(\text{external}) \cdot P(i \mid \text{external}). \quad (4)$$

Benchmarks that condition on tool use measure only the second factor. Section 6.3 shows that in the TapTax data the first factor carries most of the loss.

Status. The native judge is implemented but has not been run in production: on 6 October 2026 there are no native verdicts, so no run has a native-completion class and the Host Self-Serve Rate is unknown. What exists for every run is search behaviour, derived from stored events: whether the host searched, and, for runs with no external call, whether it answered without searching or searched and then declined. In this paper “no external tool” therefore never means “self-served”.

3.4 Eligibility

The eligible set $\mathcal{E}_c$ of intent c is the set of integrations that can genuinely serve it. It determines which runs are positive for an integration and which integrations enter a share’s denominator. The method specifies a neutral construction (InvokeRank, 2026c): a capability union over every materially relevant integration in an index, real user language with its source, intent definitions proposed by two model families and filtered by a deterministic neutrality lint, per-integration eligibility judged by both families from that integration’s own tools with quotes that must exist in its pinned snapshot, agreement-only combination, and a human review with an append-only audit log. Section F gives the full procedure.

Status. None of this has run for TapTax. The category used in this paper, `taptax-2026-10`, was written from intent definitions authored with knowledge of TapTax’s capabilities, and it is marked `instrument_validation` in the database. “Eligible” in this suite means “a positive prompt in a TapTax-authored intent”. We therefore call the baseline’s positive prompts *positive prompts (v1 labelling)* and never describe them as independently eligible. A dry run of the capability union for “UK sole trader and landlord accounting” found 18 integrations and 377 capabilities, including off-category non-UK tools (Amos, 2026, section 36); the owner has not chosen the final list.

3.5 Evidence classes and depth

Every number in InvokeRank carries an evidence class, and classes are never silently pooled: R registry data, S synthetic runs through provider APIs, H host analytics, T integration telemetry, O business outcomes, and A human-assisted runs in consumer apps typed by hand. Evidence depth is Public (R and S), Verified (sandbox Executed and Completed) or Connected (T and O). A Public score never implies execution or completion. Every S result in this paper is “API-approximated, not the consumer app”: InvokeRank never automates chatgpt.com or claude.ai, and consumer-app evidence can come only from people typing prompts by hand (A) or from telemetry (T).

4 Method

This section states method v1.0 as published (InvokeRank, 2026d) and as implemented (InvokeRank, 2026a). Where prose and code differ, the code produced the numbers, and we say so. The full derivations are in Sections A to D.

4.1 Estimation as implemented

For one integration on one surface, the published scoring path takes the positive runs of every intent and, for each stage k , proceeds as follows.

1. **Sequential counts.** For each intent cell c , (y_{ck}, n_{ck}) counts the runs that reached stage k among the runs that reached its parent.
2. **Intra-prompt correlation.** Repeats of a prompt are correlated, so ρ_{ck} is estimated by the one-way analysis-of-variance estimator for binary data with unequal group sizes, truncated to $[0, 1]$, with prompts as groups (Fleiss et al., 2003; Ridout et al., 1999). If the cell cannot estimate it (fewer than two prompts with data, no replication or no variation), the estimate pooled over all the integration’s prompts is used; failing that, a default per stage ($\rho_D = 0.6$, $\rho_S = 0.8$, $\rho_I = \rho_E = \rho_C = 0.6$).
3. **Design effect.** $\text{DEFF} = 1 + (\bar{K} - 1)\rho$, where $\bar{K} = \sum_p m_p^2 / \sum_p m_p$ is Kish’s mean cluster size over the prompts’ run counts m_p (Kish, 1965; Rao and Scott, 1992). The counts are deflated to $(y/\text{DEFF}, n/\text{DEFF})$.

4. **Shared prior.** $r_{ck} \sim \text{Beta}(\mu_k \kappa_k, (1 - \mu_k) \kappa_k)$ across the integration’s intents, fitted to the deflated counts by beta-binomial marginal maximum likelihood (Nelder-Mead on logit μ and log κ , started from the method of moments), with κ bounded between 1 and 10,000; the method of moments if the fit does not converge; Jeffreys Beta(1/2, 1/2) when fewer than two cells have trials (Robbins, 1956; Efron and Morris, 1975; Morris, 1983).
5. **Prior cap.** κ_k is capped at the deflated trials of the *other* cells; where the cap binds, μ_k becomes their pooled rate (**capPrior**). The cap was introduced on 3 October 2026 after plain plug-in empirical Bayes covered nominal 90% intervals only 35 to 48% of the time in simulation, because $\hat{\kappa}$ runs to its bound when intents look alike; plug-in intervals that treat estimated hyperparameters as known are known to undercover (Kass and Steffey, 1989).
6. **Posterior.** The conjugate update Beta($y + \mu\kappa$, $n - y + (1 - \mu)\kappa$) per stage and cell; when a parent stage never fired the child posterior is the prior, with no epsilon floor.
7. **The posterior of Q .** 4,000 draws per stage, multiplied, with an equal-tailed interval; the generator is seeded, so the output is deterministic.
8. **Displayed proportions.** Single stage rates are shown with Jeffreys intervals on the deflated counts (Jeffreys, 1946; Brown et al., 2001).
9. **Roll-up.** Across intents, $\sum_c w_c Q_c$ with stated weights, equal until demand data exist; draws are combined index by index, on the assumption that intents are independent.
10. **Interval levels and ties.** Public monitoring uses 90% intervals; experiment ship decisions and the placebo and blinded controls use 95%. Overlapping 90% intervals are displayed as “intervals overlap, reported as tied”, which is a display rule, not an equivalence test.

We wrote an independent Python implementation of this path, porting the random number generator, the optimiser and the special functions line by line (**scripts/ir_paper/method.py**). From the per-run records alone it reproduces the production reports’ headline and every intent interval on both surfaces to floating-point precision, with a largest absolute difference below 10^{-13} points (Section 7.1).

4.2 Derived metrics

Loss share. $L_k = -\ln \tilde{r}_k / \sum_j -\ln \tilde{r}_j$ with posterior means $\tilde{r}$. Because $-\ln Q = \sum_k -\ln r_k$, the shares decompose the log loss exactly and sum to one. They are diagnostic, not causal. A stage at rate 1 contributes nothing; when every stage is near 1 the denominator approaches 0 and the shares become unstable; a stage at rate 0 takes the whole loss (Section B).

Lever. $\text{Lever}_k = \tilde{Q} (r_k^{\text{ref}} / \tilde{r}_k - 1)^+$, where r^{ref} is the intent median or the best competitor’s rate. It is the algebraic change in Q if stage k alone moved to the reference with every other stage fixed: potential uplift, not a treatment effect (Section C). Runs judged as correct native self-service are excluded from the Lever unless the customer declares tool use necessary.

False-Invoke Rate. Calls to the integration on negative prompts divided by negative runs, with a Jeffreys interval on counts deflated at the pooled ρ_S . It guards against buying recall with indiscriminate invocation. Note the definition: the published path counts calls to the *focal* integration only. A host can also call a competitor on a negative prompt; we report that separately as “any external call”.

Reliability. pass³ per prompt, estimated as $\binom{c}{3}/\binom{n}{3}$ from c successes in n repeats, the unbiased pass^k estimator of τ -bench (Yao et al., 2025), a re-application. With $K = 2$ in the TapTax baseline it cannot be computed there.

Invoke Share and Selection Share. For a contested intent c (at least two independently eligible integrations), $IS_{ic} = Q_{ic} / \sum_{j \in \mathcal{E}_c} Q_{jc}$. The no-tool path and host built-in tools are never in the denominator. When any eligible integration lacks Executed or Completed, every intent is computed at $D \times S$ and the metric is labelled Selection Share, so a roll-up never mixes the two (Section D). It is not market share and not share of all resolutions; the eligibility set defines the denominator. Independent per-integration posterior draws do not preserve the compositional dependence of first choices: within a run, first choices are mutually exclusive, so the true Q_{ic} are negatively dependent, and independent draws overstate the variance of the shares’ sum and misstate each share’s interval. **Status:** not computed for TapTax, because no neutral eligibility exists.

Invoke Rating. A conditional logit over first choice (a multinomial Bradley-Terry model) with an outside option of utility 0 for “no tool”, a position term γ and a branded-prompt term λ , intent effects partially pooled toward the integration’s β with a Gaussian prior, fitted by penalised batch Newton-Raphson, and displayed as $R = 1000 + 400 \beta / \ln 10$, so that $R = 1000$ means “as likely to be chosen as no tool at all” (McFadden, 1974; Bradley and Terry, 1952; Elo, 1978; Chiang et al., 2024). Intervals come from a bootstrap over prompts (200 resamples by default). It is implemented and fixture-tested but **has never been fitted to TapTax data**. Two tensions deserve stating. Invoke Rating includes the outside option, while Invoke Share excludes it, so the two can move in opposite directions when the host’s own answer gains or loses ground. And the conditional logit assumes independence of irrelevant alternatives, which fails for near-duplicate tools: adding a clone of an integration should split its share, not draw proportionally from every option; the PRD proposes a nested logit for this case (Luce, 1959; McFadden, 1978; Train, 2009).

4.3 What the paper examines critically

Posterior independence is a property of the model, not an extra assumption. Under independent runs with independent stage priors, the sequential-binomial likelihood factorises in $(r_D, \dots, r_C)$, so the stage posteriors are exactly independent (Section A). The real question is misspecification. When prompts are the sampling units, prompt-level heterogeneity can make the estimators of r_D and r_S covary, and multiplying independent draws can then misstate the variance of Q . Section 7.5 measures this with a prompt-cluster bootstrap.

Per-cell ρ is unstable at $K = 2$. With two repeats per prompt, the ANOVA estimator in a cell of about two dozen prompts often lands on a truncation bound. $\rho = 0$ means no deflation and intervals that are too narrow; $\rho = 1$ means each prompt counts once. Section 7.4 shows how often this happens.

The prior cap is an engineering regulariser. It was motivated by a simulation failure and is not a derived result; the principled corrections are a Kass-Steffey adjustment or a full hierarchical Bayes fit (Kass and Steffey, 1989; Gelman et al., 2013). Its coverage claim (86 to 90% at 30 prompts and $K = 3$; 90% at 100 prompts) comes from a test whose data-generating process is narrow: three intents with identical true stage rates, prompt-level heterogeneity with $\kappa = 3$, $K = 3$ (InvokeRank, 2026a, `invokerank.test.ts`). Section 7.5 reports a wider simulation.

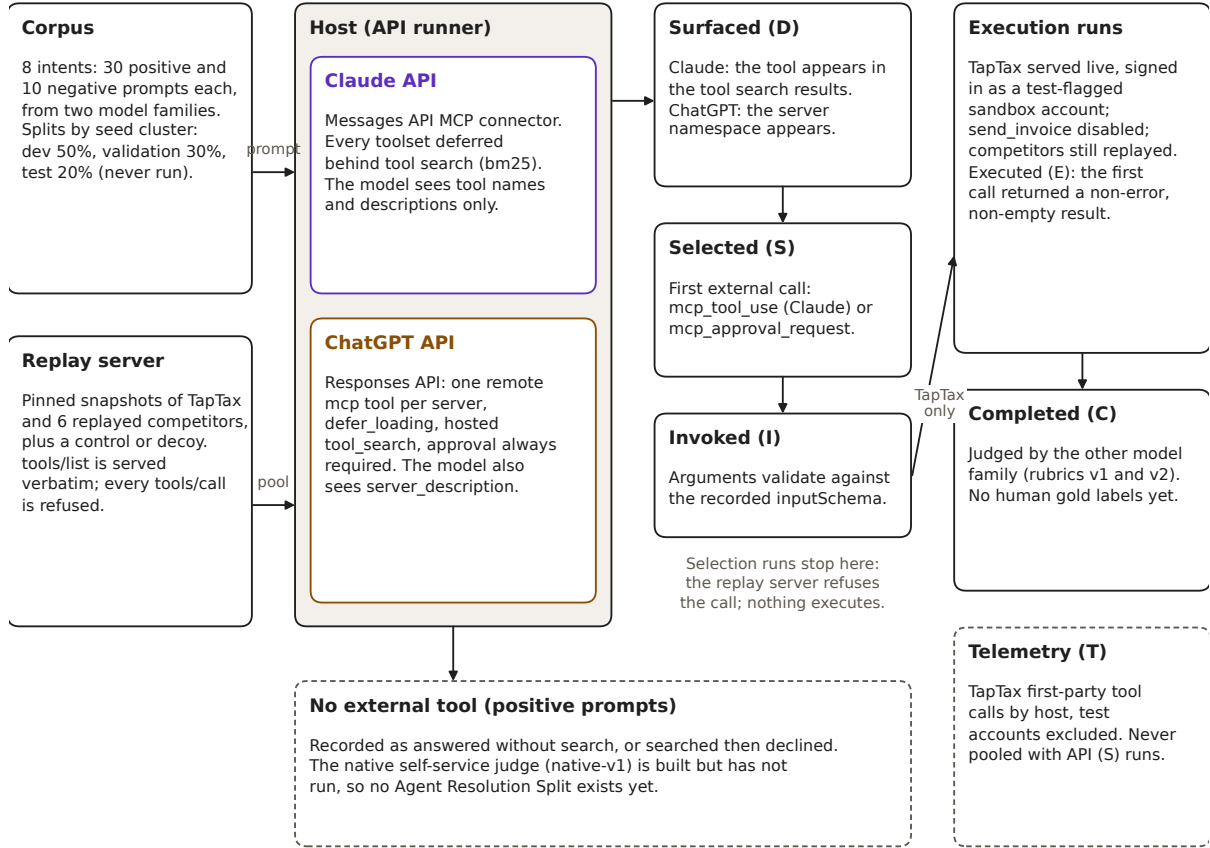


Figure 1: Measurement architecture as built for the TapTax programme. Selection runs observe Surfaced, Selected and Invoked against a replayed, pinned pool; execution runs serve TapTax live from a sandbox account and add Executed and a cross-family judged Completed; runs with no external call form the no-tool branch. Where each stage is observed differs by host (Table 3). Schematic; no data.

Runs as a level of clustering. A run session or batch may share provider-side state, so identical arms run at different times may differ by more than prompt-level intervals imply. This would violate the exchangeability that both the model interval and a prompt-only bootstrap assume. Section 7.3 tests for it directly.

The roll-up assumes independent intents. With a shared prior the intents are not independent a posteriori, and when the cap binds each intent’s posterior is nearly the pooled one; the roll-up interval ignores this dependence (Section 7.5).

5 Experimental setup

Figure 1 shows the measurement architecture as built.

5.1 TapTax, the owned proof environment

TapTax is Making Tax Digital (MTD) software for UK sole traders and landlords: it keeps income and expense records, invoices and mileage, estimates tax and prepares quarterly updates for HM Revenue and Customs. It exposes a production remote MCP server with OAuth 2.1 and is listed in the official MCP Registry; InvokeRank measures it through a separate private measurement entry, so the measured identity stays stable when the public listing changes. The

same individual owns TapTax and InvokeRank. TapTax is used here because it is the one integration where InvokeRank has ground truth on every stage: a live sandbox account for execution, first-party telemetry, and the ability to make byte-identical, blinded and relabelled copies of the server. It is used to find where the instrument is wrong. It is not evidence that the method generalises to other categories, and it is not evidence that TapTax is better than its competitors.

Versions. Every experiment pins the exact server snapshot it replays. The 3 October baseline, pilot, execution runs, metadata experiment and control round use TapTax 1.0.0 with 15 tools. Version 1.1.0, released on 4 October, added calculator tools and lists 48; later arms pin 1.0.0, 1.1.0, a later 1.1 release (“1.1 live”) or measurement-only variants (Section I).

Competitors. The pool holds TapTax and 6 competitors approved by the owner, all public registry listings, all replayed from recorded snapshots and never called. Their eligibility per intent has not been independently established, so their raw scores stay internal (Amos, 2026, section 36, item 11). In this paper they appear only as one pooled class, “a competitor”, and no per-competitor score is reported.

5.2 Corpus and splits

The MTD corpus `taptax-2026-10` has 8 intents and 320 prompts: 240 positive (including 19 where the right behaviour is to ask a clarifying question) and 80 negative prompts, where the right behaviour is to use no tool. Claude Sonnet 5.5 wrote 160 prompts and GPT-6.1 Sol wrote 160, from the intent definitions, with embedding de-duplication at cosine 0.9, a brand-name exclusion list and canary strings. Prompts are grouped into 208 seed clusters, and splits are made by cluster so that paraphrases never straddle splits: development (182 prompts), validation (79) and a locked test split (59: 45 positive and 14 negative). **The locked test split has never been run.** It is available for a pre-registered confirmatory analysis (Section 9).

Table 2 lists the eight intents with their stored definitions and success criteria. The intent definitions were written with knowledge of TapTax’s capabilities; this is the v1 eligibility problem of Section 3.4. The need-type column is our own provisional coding from the definitions and stored criteria, by one coder who is not independent of the authors.

Corpus realism. A deterministic phrase rule flags prompts that depend on context the harness never supplies. Of the 320 prompts, 37 are flagged (31 mention an attachment, 4 a document, 2 a screenshot, 2 “this record” and 1 “the above”; a prompt can carry several flags). The human review of the 91-prompt sample (59 stratified and the rest flagged) has not been done (0 prompts reviewed). These prompts are therefore still in every figure reported here.

Negative labels. The method allows a negative prompt to be relabelled *optional* (counted as neither positive nor negative) when both model families independently judge, from the prompt text alone, that using an app would be reasonable. This relabel has run on the calculator corpus (32 negatives became optional) but not on the MTD corpus, so MTD false-invoke rates exist only on the original labels.

5.3 Hosts and runners

Table 3 summarises the two runners as implemented and observed (InvokeRank, 2026c,a). Every run logs the requested model and the model string the provider returned; every baseline run returned `claude-sonnet-5-5` or `gpt-6.1-sol`. Runs use each surface’s default reasoning effort, no `temperature` (Sonnet 5.5 rejects non-default values, and the method forbids setting it on measured runs) and no refusal fallback, because a fallback answers with another model.

Two consequences follow. First, r_D and r_S are not comparable across hosts one at a time, because Surfaced means a tool on one host and a server on the other; only $r_{D r_S}$ is. Second, the levers available to an integration differ by host: on the Claude API path only tool names and

Table 2: The eight intents of the MTD corpus, with the stored definition and success criterion (InvokeRank, 2026b). Need type, read (R) or write (W), and whether native self-service is plausible are a provisional single-coder classification (Section 5.2). The stored criterion of `mtd-deadlines` asks for “their” dates, which a generic answer cannot satisfy without the person’s accounting period.

Intent	Definition (stored)	Success criterion (stored)	Need type	R/W	Native
Check MTD obligation <code>check-mtd-obligation</code>	Find out whether Making Tax Digital for Income Tax applies to me, from which date, based on my self-employment and property income.	A clear yes or no with the start date that applies to their income level.	answerable	R	yes
MTD deadlines <code>mtd-deadlines</code>	Know the dates of my upcoming quarterly updates and year-end submissions so I do not miss a deadline or get a penalty.	The next due dates for their quarterly updates and final declaration.	private state (contested)	R	partly
Prepare quarterly update <code>prepare-quarterly-update</code>	Get my quarterly update for this quarter ready to send to the tax office, checking the totals before anything is submitted.	The quarter’s income and expense totals are prepared for review, with nothing filed without their confirmation.	private state	R	no
Record a transaction <code>record-transaction</code>	Record a piece of business income or an expense (from a receipt, bank line or memory) in my books with the right category.	The transaction is saved in their records with amount, date and category.	action	W	no
Create and send invoice <code>create-send-invoice</code>	Create an invoice for work I have done for a client and get it sent to them, then keep track of whether it has been paid.	An invoice exists with the right client, lines and total, and is sent or ready to send.	action	W	no
Log business mileage <code>log-business-mileage</code>	Log the business journeys I drove and see how much I can claim in mileage allowance for the tax year.	The trip is logged and the claimable amount at the approved mileage rates is shown.	action	W	no
Estimate tax and NI <code>estimate-sole-trader-tax</code>	Work out roughly how much income tax and National Insurance I will owe on my self-employed profit this tax year, so I can put the right amount aside.	A figure for income tax and Class 2/4 National Insurance for the stated profit and tax year, with the main allowances applied.	answerable	R	yes
Check tax code <code>check-tax-code</code>	Check whether the tax code on my payslip or letter looks right for my situation, and what it means for the tax taken from my pay.	An explanation of the code with whether it is likely correct for their stated circumstances.	answerable	R	yes

descriptions can steer selection, while on the OpenAI path the server description is an additional lever. Section 6.7 shows that this difference is not academic.

5.4 Replay server, rotations and selection runs

A replay server serves a recorded snapshot of each integration’s `tools/list` and server metadata verbatim and refuses every `tools/call`. A selection run sends one prompt to the host with the pool of competing servers, all from the replay server, so the pool is byte-identical across runs whatever the live servers do. The first tool call is the selection; nothing executes. Tool order is a nuisance variable: each run uses one of 5 fixed cyclic rotations of the pool, balanced across prompts. Competitors are only ever replayed, so competitor evidence stops at Selected.

5.5 Execution runs and the judge

In an execution run the focal integration is served live (production TapTax), signed in as a test-flagged sandbox account with paid features, with OAuth tokens sealed at rest; competitors stay on the replay server. `send_invoice` is disabled, so no email reaches an invented client, and negative prompts are never executed. Executed is deterministic: the first call returned a non-error, non-empty result. Completed is judged by a model from the *other* family (GPT-6.1 Sol judges Claude API runs and Claude Sonnet 5.5 judges ChatGPT API runs) against a binary

Table 3: The two host surfaces as built. “Surfaced” is observed at tool level on Claude and at server level on ChatGPT, so stage-wise host comparisons of r_D and r_S are not like-for-like; the product $r_D r_S$ is.

	Claude API	ChatGPT API
Adapter and runner model	<code>anthropic_messages</code> ; Claude Sonnet 5.5	<code>openai_responses</code> ; GPT-6.1 Sol
Tool access	Messages API MCP connector (beta <code>mcp-client-2026-09-15</code>); every toolset deferred behind tool search, bm25	One remote mcp tool per server with <code>defer_loading</code> , hosted <code>tool_search</code> , <code>require_approval: always</code>
Server metadata the model sees	Tool names and descriptions; no server description is sent, and <code>initialize.instructions</code> did not reach the model in a recall probe (4 Oct)	Tool names and descriptions plus the caller-set <code>server_description</code> ; <code>initialize.instructions</code> also not visible
Surfaced (D) observed as Selected (S) observed as Invoked (I)	the tool in <code>tool_search_tool_result.tool_references</code> , named <code><server>_<tool></code> first <code>mcp_tool_use</code> selected, with arguments that validate against the recorded <code>inputSchema</code>	the namespace <code>mcp_<server></code> in <code>tool_search_output.tools</code> first <code>mcp_approval_request</code> (or <code>mcp_call</code>) same
Sampling	default effort; no <code>temperature</code> ; no refusal fallback	default effort; no <code>temperature</code> ; no <code>seed</code>
Batch	Message Batches for some suites, at half price	synchronous only

checklist built from the intent’s success criteria, with evidence quotes. Rubric `completion-v2` was introduced after the owner and the agent working on TapTax reviewed the `completion-v1` verdicts; it gives the judge each prompt’s expected behaviour and counts “justified stops” (confirming an irreversible action, or naming information the request and account genuinely lack). Both verdicts are retained. We treat the rubric revision as a researcher degree of freedom and report completion under both rubrics. The planned gold set has 300 human labels, with the judge trusted at Cohen’s $\kappa \geq 0.70$; on 6 October 2026 it holds 0 labels.

5.6 Telemetry

A read-only view in TapTax’s database exposes real tool calls (tool, client host, outcome, duration and hour, with no user identifiers and with test accounts excluded); InvokeRank reads it hourly. These are T evidence and are never pooled with the API runs.

5.7 The programme, its cost and the data

Between 3 and 5 October 2026 the programme recorded 26,886 TapTax runs in 136 suites (14,085 on the ChatGPT API and 12,801 on the Claude API), of which 20,540 completed, 4,758 failed and 1,588 were still pending at export. Table 5 groups them into 21 families and states which appear in the body of the paper and which only in Section I; the garden of forking paths is large, and the ledger is there so the reader can count it. The core programme cost \$5.72 for the pilot, \$16.28 for the baseline, \$10.83 for the same-day repeat, \$5.81 for the execution runs, \$6.35 for the metadata experiment and \$15.64 for the control round, in model spend recorded per run. A synchronous Claude API selection run cost \$0.0246 on average, a batched one \$0.0130 and a ChatGPT API run \$0.0078; no run read anything from a prompt cache (0 cached input tokens in total).

The data pack is a read-only export taken on 6 October 2026 with no model calls (InvokeRank, 2026b). It holds per-run records, prompt metadata, intent definitions, suite pools with pinned versions and tool counts, judgements under both rubrics, control verdicts with their full metrics, costs and telemetry, and the published path’s own reports. Prompt text is deliberately not exported because the corpus is private. Every number in this paper is computed from the pack by the scripts in the paper’s repository directory and carries a provenance entry in a generated ledger (Section 11); Table 4 gives the evidence class and source of the central ones.

Table 4: Evidence provenance of the paper’s central numbers. Class: S synthetic API runs, T integration telemetry; derived: computed from S numbers; SIM: the simulation of Section H; PRD: the internal requirements document, reported where the data do not reproduce it. Source codes: [DB] the data pack exported from production on 6 October 2026, [CODE] the implementation. Every [DB] number is reproduced by `scripts/analyse.py` and `scripts/build_numbers.py` from `runs.csv` and its siblings.

Number	Class	Source and filter
Selection-level InvokeRank, 41.5 and 38.3	S	[DB] <code>taptax-uk</code> , measurement, selection, dev and validation, 3 Oct; published path (exact port)
No external tool, 57.4% and 57.2%	S	[DB] as above; <code>selected_role</code> , <code>no_tool_reason</code>
Completed, 57.9% and 42.6% (v2)	S, judged	[DB] <code>taptax-uk</code> , execution, dev; <code>judgements.csv</code> ; uncalibrated
Control verdicts (Table 10)	S	[DB] <code>taptax-uk-ctl-2026-10-04-*</code> , validation; <code>checks.csv</code>
Server-description lifts, +42.0 and +41.3	S	[DB] <code>taptax-x-*-desc-*</code> against live replicates, validation; run-and-prompt bootstrap
Between-run test, $p = 0.014$	S, derived	[DB] replicate groups of Section 7.3; within-prompt permutation
Interval coverage, 87.1% and 82.2%	SIM	<code>data/sim/coverage.csv</code> ; scoring package via harness
Real tool calls, 0 and 8	T	[DB] <code>telemetry.csv</code> , 90 days to 5 Oct
Pilot ρ , 0.71 to 0.85	PRD	Reported, not reproduced from [DB] (Section 7.1)
Default ρ , 0.6 and 0.8	[CODE]	<code>packages/scoring/src/deff.ts</code>

6 Results

All results in this section are S evidence: API-approximated, not the consumer app. Unless stated otherwise they come from suite `taptax-uk`, purpose `measurement`, run type `selection`, complete runs of 3 October 2026, on the development and validation splits, with $K = 2$ repeats and 5 rotations; intervals are 90%.

6.1 Baseline: the funnel by host

On each surface the baseline ran 195 positive prompts (v1 labelling) and 66 negative prompts twice, 522 runs per surface. TapTax was surfaced in 265 of 390 positive Claude API runs and selected in 162; on the ChatGPT API it was surfaced in 260 and selected in 149 (Figure 2, Table 6). Every selected call had valid arguments, so Invoked equalled Selected on both surfaces. The Selection-level InvokeRank was 41.5 (37.1 to 46.1) on the Claude API and 38.3 (34.5 to 42.1) on the ChatGPT API. The two intervals overlap, so by the method’s display rule the hosts are reported as tied. The pilot earlier the same day, on 40 positive prompts with $K = 3$ (176 runs per surface including negatives), gave 43.9 (35.8 to 52.1) and 44.7 (37.3 to 52.5).

Two stages carry all the loss. On the Claude API the log-loss shares from pooled rates were 44.0% at Surfaced and 56.0% at Selected; on the ChatGPT API 42.1% and 57.9%. Because Surfaced is observed at tool level on Claude and at server level on ChatGPT, these splits are not comparable across hosts; the product r_{DRS} is.

6.2 Hosts and intents

The headline averages over very different intents (Figure 3, Table 7). Preparing a quarterly update scored 75.9 (62.8 to 87.8) on the Claude API and 57.4 (44.5 to 70.0) on the ChatGPT API. Estimating income tax and National Insurance scored 17.7 (8.3 to 30.1) and 3.5 (0.4 to 9.1), and checking a tax code 6.2 (1.1 to 15.2) and 1.5 (0.0 to 5.5). The hosts disagree most on whether MTD applies: 44.9 (29.8 to 60.5) on the Claude API against 66.6 (54.2 to 78.3) on the ChatGPT API, with no overlap.

Table 5: Experiment ledger: every suite family run on TapTax, 3 to 5 October 2026, including abandoned and incomplete suites. Hosts: C is the Claude API, G the ChatGPT API. Dates are October 2026 (UTC). Corpus: MTD is `taptax-2026-10`, calc the calculator corpus. Batched: runs sent through Anthropic Message Batches. Cost: model spend recorded on the runs, US dollars. Source: `runs.csv` and `members.csv` (InvokeRank, 2026b).

Family	Suites	Hosts	Dates	Pinned versions	TapTax Corpus	Runs	Complete	Failed	Pending	Batched	Cost (\$)	In paper
Pilot, baseline and execution (taptax-uk)	1	C+G	3 Oct	1.0.0 (15 tools)	MTD	1,670	1,670	0	0	58	27.81	Sections 6 and 7
Same-day repeat	1	C+G	3 Oct	1.0.0 (15 tools)	MTD	1,044	1,044	0	0	522	10.83	Section 6
Metadata experiment (3 Oct)	2	C+G	3 Oct	1.0.0 (15 tools)	MTD	632	632	0	0	316	6.35	Section 6
Instrument controls, round 2026-10-04	5	C+G	4 to 5 Oct	1.0.0 (15 tools)	MTD	1,580	1,580	0	0	790	15.64	Section 7
Version watch: 1.0.0 against 1.1.0	4	C+G	4 Oct	1.0.0 (15 tools); 1.1.0 (48 tools)	MTD	1,264	1,264	0	0	474	14.48	Appendix I
Version watch: later MTD comparisons	4	C+G	4 to 5 Oct	1.1.0 (48 tools)	MTD	1,264	1,264	0	0	631	12.62	Appendix I
Calculator corpus baselines	3	C+G	4 Oct	1.1.0 (48 tools)	calc	428	428	0	0	214	4.93	Appendix I
Abandoned comparisons (void)	6	C+G	4 Oct	1.1.0 (48 tools)	calc	2,568	195	2,373	0	0	1.89	Appendix I
Version watch: calculator comparisons	16	C+G	4 to 5 Oct	1.1.0 (48 tools)	calc	2,240	2,240	0	0	1,120	25.10	Appendix I
Tool-inventory arms (keep15, keep23)	2	C	4 Oct	1.0.0 (15 tools); 1.1.0 (23 tools)	MTD	316	316	0	0	316	3.95	Appendix I
Split-server arms	2	C	4 Oct	1.0.0 (15 tools); 1.1.0 (33 tools)	MTD	316	316	0	0	316	3.94	Appendix I
Account-wording arms	3	C+G	4 Oct	1.1.0 (48 tools)	MTD	474	474	0	0	158	4.39	Appendix I
Disambiguation candidate (1.1.1)	1	C	4 Oct	1.1.0 (48 tools)	MTD	158	158	0	0	158	1.88	Appendix I
Replicates of the live version	24	C+G	4 to 5 Oct	1.1.0 (48 tools)	both	3,904	3,084	184	636	1,316	30.39	Appendix I
Replicates of 1.0.0	12	C	5 Oct	1.0.0 (15 tools)	MTD	1,896	935	11	950	946	11.74	Appendix I
Server-description arm	18	G	4 to 5 Oct	1.1.0 (48 tools)	both	2,324	1,051	1,273	0	0	7.85	Appendix I
Second server-description	8	G	5 Oct	1.1.0 (48 tools)	both	744	743	1	0	0	5.56	Appendix I
wording												
Calculator facts plus server description	8	G	4 to 5 Oct	1.1.0 (48 tools)	both	744	744	0	0	0	5.71	Appendix I
Calculator facts in tool descriptions	8	C+G	4 to 5 Oct	1.1.0 (48 tools)	both	1,488	1,486	0	2	742	16.69	Appendix I
Misroute-fix descriptions	6	C+G	5 Oct	1.1.0 (48 tools)	both	1,200	600	600	0	600	8.69	Appendix I
Mileage wording	2	C+G	4 to 5 Oct	1.1.0 (48 tools)	MTD	632	316	316	0	0	2.39	Appendix I

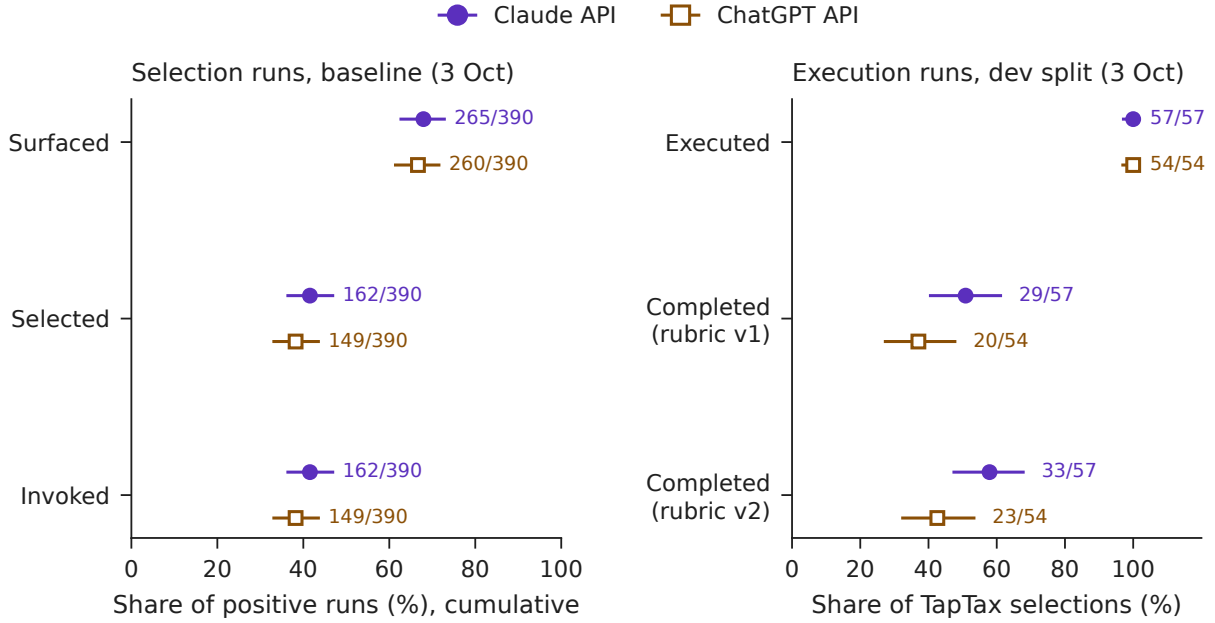


Figure 2: The funnel by host. Left: cumulative share of positive baseline runs (3 October, `taptax-uk`, measurement, selection, dev and validation splits; 195 prompts, 390 runs per host) reaching Surfaced, Selected and Invoked; 90% Jeffreys intervals on counts deflated by the intra-prompt correlation of each cumulative event. Right, on its own denominator: of TapTax selections in the execution runs (dev split, $K = 1$), Executed and Completed under rubrics v1 and v2; 90% Jeffreys intervals. The two panels never share a denominator. Surfaced is tool-level on the Claude API and server-level on the ChatGPT API. Evidence class S.

Selection and what the task needs (RQ4). The pattern is suggestive: the intents that need the person’s own records or an action score higher than those a host can plausibly answer from knowledge. Under the provisional coding of Table 2, the mean intent score of the five tool-need intents exceeded that of the three answerable intents by 29.7 points on the Claude API and 23.0 points on the ChatGPT API. With eight intents, a prompt-level model with intent as a random effect has only eight units at the level that matters, so we test at the intent level directly: an exact one-sided permutation test over the 56 ways of choosing five of eight intents gives $p = 0.036$ on the Claude API and $p = 0.107$ on the ChatGPT API. The result depends on the coding: if `mt-d-deadlines` is coded answerable (generic statutory dates), the p -values become 0.114 and 0.257. Three further caveats apply. There are only eight intents; need type is confounded with prompt wording and with TapTax’s own tool descriptions; and `check-mtd-obligation`, which a host can answer from knowledge, scores 66.6 on the ChatGPT API. We read this as a hypothesis for the native judge and an independent coding to test, not as a finding.

6.3 The host as a competitor

On both surfaces most positive runs ended with no external tool: 224 of 390 (57.4%) on the Claude API and 223 of 390 (57.2%) on the ChatGPT API (Figure 4). A competitor was the first call in only 4 Claude API runs and 18 ChatGPT API runs. The alternative to TapTax was overwhelmingly the host itself, not another integration.

The hosts got there differently. The Claude API answered 119 positive runs without searching for a tool at all and searched then declined in 105. The ChatGPT API answered only 37 without searching and searched then declined in 186. In the two-level model of Equation (4), the Claude API more often decides at the first level not to look, and the ChatGPT API more often looks and

Table 6: Stage results by host, baseline. y of n : runs reaching the stage among runs that reached its parent. Rate: pooled y/n ; interval: 90% Jeffreys on counts deflated at the pooled ρ . ρ is pooled over the prompts of all intents (the published fallback); Invoked has no variation, so its ρ is the default. Loss share from the pooled rates. The last row of each host is the published headline (posterior roll-up over intents, 90% credible interval).

Host	Stage	y	n	Rate (%)	90% interval	ρ (source)	DEFF	n_{eff}	Loss share (%)
Claude API	Surfaced (D)	265	390	67.9	62.4to 73.2	0.94(pooled)	1.94	200.9	44.0
	Selected ($S \mid D$)	162	265	61.1	54.4to 67.5	0.81(pooled)	1.79	147.9	56.0
	Invoked ($I \mid S$)	162	162	100.0	98.2to 100.0	0.6(default)	1.55	104.6	0.0
	<i>Selection-level InvokeRank</i>	162	390	41.5	37.1to 46.1				
ChatGPT API	Surfaced (D)	260	390	66.7	61.1to 71.9	0.91(pooled)	1.91	204.4	42.1
	Selected ($S \mid D$)	149	260	57.3	50.4to 64.0	0.87(pooled)	1.84	141.0	57.9
	Invoked ($I \mid S$)	149	149	100.0	98.0to 100.0	0.6(default)	1.56	95.8	0.0
	<i>Selection-level InvokeRank</i>	149	390	38.3	34.5to 42.1				

Table 7: Intent results, baseline. Score: Selection-level InvokeRank with 90% credible interval. No search: positive runs with no external call and no tool search. Declined: positive runs where the host searched and called nothing. FIR: TapTax calls over negative runs on the original labels (the MTD corpus has not been relabelled). Intents ordered by the Claude API score.

Intent	Prompts	Claude API				ChatGPT API			
		Score (90%)	No search	Declined	FIR	Score (90%)	No search	Declined	FIR
prepare-quarterly-update	23	75.9(62.8to 87.8)	0	9	0/18	57.4(44.5to 70.0)	0	19	0/18
mtd-deadlines	23	56.5(41.4to 70.8)	6	11	0/18	60.2(47.0to 72.3)	3	13	0/18
record-transaction	30	52.7(40.2to 65.7)	3	24	0/16	45.9(33.9to 58.0)	3	27	0/16
log-business-mileage	24	45.5(30.7to 60.5)	10	15	0/14	33.6(23.2to 45.2)	5	30	0/14
check-mtd-obligation	22	44.9(29.8to 60.5)	23	1	0/16	66.6(54.2to 78.3)	1	4	4/16
create-send-invoice	25	32.3(19.4to 46.8)	2	33	0/16	37.4(25.7to 50.2)	5	26	0/16
estimate-sole-trader-tax	27	17.7(8.3to 30.1)	43	2	0/16	3.5(0.4to 9.1)	5	46	1/16
check-tax-code	21	6.2(1.1to 15.2)	32	10	0/18	1.5(0.0to 5.5)	15	21	0/18
All intents	195	41.5(37.1to 46.1)	119	105	0/132	38.3(34.5to 42.1)	37	186	5/132

then decides against every candidate. The two behaviours call for different interventions: a host that never searches cannot be reached through tool metadata at all, while a host that searches and declines has read the metadata and rejected it, or preferred its own answer. Whether the host’s own answers met the need is unknown: the native judge has not run, so the Host Self-Serve Rate cannot be reported.

A degenerate published split. Run on the baseline today, the published path’s Agent Resolution Split excludes every unjudged no-tool run from its denominator, as specified. With zero native verdicts, it therefore classifies only the runs with an external call: 166 runs on the Claude API, with TapTax’s class at 98.0% after intent weighting, and 224 runs listed as unjudged; on the ChatGPT API 167 runs, 78.8% and 223 unjudged. The report labels the unjudged count, but a reader who looks only at the class shares would see TapTax “completing” almost every eligible request. The publication gates refuse such a snapshot; the internal report does not.

6.4 Execution versus completion

TapTax was selected in 57 of 137 positive Claude API execution runs and 54 of 137 ChatGPT API runs on the development split, and every selected call executed without a tool error (Table 8). Executed is therefore complete on both surfaces, while judged completion was far lower. Under rubric v2 the judge counted 33 of 57 Claude API runs completed (57.9%, 47.0 to 68.2) and 23 of 54 ChatGPT API runs (42.6%, 32.0 to 53.8). Under rubric v1 the same runs gave 29 (50.9%) and 20 (37.0%). The revision, made after inspecting v1 verdicts, turned 6 Claude API failures into completions and 2 completions into failures, and 5 and 2 on the ChatGPT API.

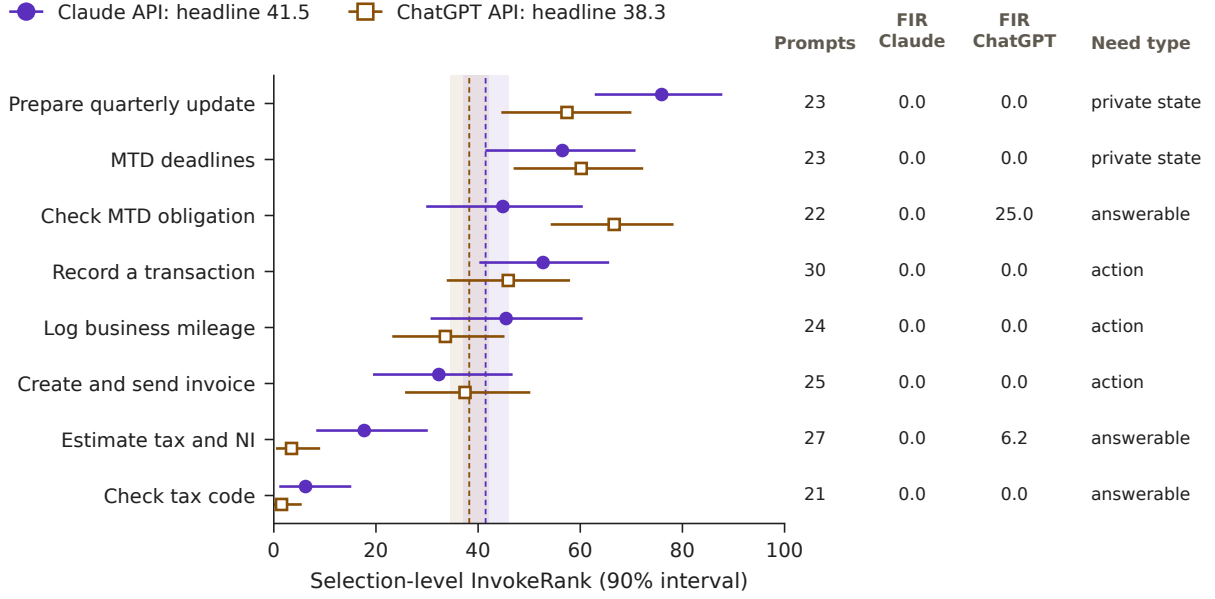


Figure 3: Intent-level Selection-level InvokeRank, baseline, with 90% credible intervals from the published path; the vertical bands are each host’s headline and interval. Text columns: unique positive prompts per intent (`runs.csv`), TapTax false-invoke rate on the intent’s negative runs (%), and the provisional need type of Table 2. Filter as Figure 2. Evidence class S.

The failures concentrate in two places. Invoice creation completed in 2 of 11 runs across both hosts: the models listed existing invoices and stopped, although the request held enough to draft one. On the ChatGPT API, MTD deadlines completed in 0 of 7 runs: the tool returned the dates, and the final answer did not relay them. Both failure modes sit in the host’s orchestration and final response, not in TapTax’s server, which answered every call. The trajectories themselves are not in the data pack (it holds no answer text), so these two descriptions rest on the review of the runs recorded in the PRD (Amos, 2026, section 36). Executed therefore does not predict Completed here (RQ6), and a stage that tool-use benchmarks often treat as the end of the task is, for this integration, roughly the midpoint.

The host difference is confounded with the judge. Claude API runs are judged by GPT-6.1 Sol and ChatGPT API runs by Claude Sonnet 5.5. The difference between 57.9% and 42.6% cannot be read as a host effect until a gold set calibrates each judge separately, or both judges score both hosts. With 0 gold labels, Cohen’s κ , the judge’s true and false positive rates and the prediction-powered correction $\hat{\theta} = \bar{J}_N + \overline{(H - J)}_n$ (Angelopoulos et al., 2023) are all pending, and Completed is the judge’s uncorrected rate.

6.5 False invokes

On negative prompts, where the right behaviour is no tool, the Claude API called TapTax in 0 of 132 runs and called no external tool at all (0 runs). The ChatGPT API called TapTax in 5 of 132 negative runs (3.8%, upper 90% bound 9.0%) and made some external call in 11. Of these, 4 were on negative prompts of the intent asking whether MTD applies, out of 16 such runs. In the pilot the ChatGPT API called TapTax on 5 of 6 such negative runs. Two definitions are in play here, and earlier summaries conflated them: the published False-Invoke Rate counts calls to the focal integration (5 of 132), while “false invokes” in the earlier working summaries counted any external call (11 of 132).

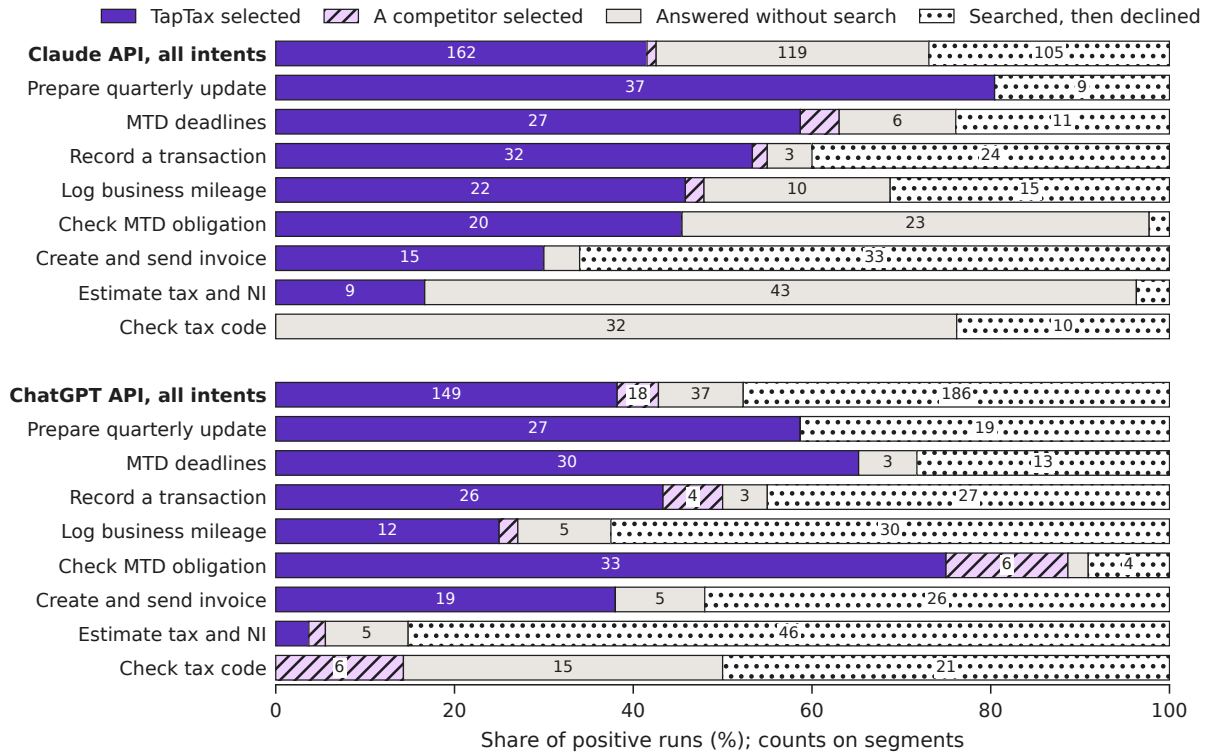


Figure 4: What happened on positive baseline runs, per host and per host and intent; counts on segments. These are search behaviours, not success classes: no native self-service verdict exists yet, so “answered without search” does not mean the need was met. Competitors are pooled and anonymised. Filter as Figure 2. Evidence class S.

6.6 Same-day repeat

A full repeat of the baseline ran the same afternoon on both surfaces. On the Claude API it scored 44.1 (39.8 to 48.5) against 41.5 (37.1 to 46.1), and on the ChatGPT API 39.0 (35.1 to 42.9) against 38.3 (34.5 to 42.1). The headline intervals overlap on both surfaces, as do the intent intervals (8 of eight on the Claude API and 8 of eight on the ChatGPT API); overlap is the method’s stability rule. The paired comparison is less reassuring on the Claude API. Paired over the same 195 prompts, the change in TapTax selection was +2.6 points (90% prompt bootstrap +0.3 to +4.9), an interval that excludes zero; on the ChatGPT API the paired change was 0.0 (−2.1 to +1.8). The Claude API repeat also changed the execution path: all its runs went through the Batch API, while 58 of the baseline’s 522 did (a batch lost most of its requests to a rate limit and they were re-run synchronously). A batch-path effect, a time effect and a one-in-ten chance exceedance cannot be separated with one pair, and the overlap rule would not have noticed any of them. This is same-day repeatability, not temporal stability; the week-apart repeat scheduled for 10 October has not run.

6.7 Metadata experiments

The 3 October tool-description candidate. The agent working on TapTax rewrote three tool descriptions, those of `estimate_self-employed_tax`, `check_tax_code` and `create_invoice`, and the candidate was paired against the current version on the validation split (58 positive prompts, $K = 2$, both surfaces). Selection moved from 40.5% to 40.5% on the Claude API, a paired change of 0.0 points (−6.0 to +6.0), and from 31.0% to 34.5% on the ChatGPT API, +3.4 (−0.9 to +8.6); TapTax calls on negatives stayed at 0 of 42 in every arm (Figure 5, left). This is a null result at the experiment’s resolution: with the observed spread of per-prompt differences,

Table 8: Execution and completion, development split, 3 October ($K = 1$). Selected: positive runs whose first call went to TapTax. Executed: the first call returned a non-error, non-empty result. Completed: the cross-family judge’s verdict (GPT-6.1 Sol for Claude API runs, Claude Sonnet 5.5 for ChatGPT API runs), uncorrected (0 human gold labels), with 90% Jeffreys intervals for rubric v2.

Host (judge)	Intent	Selected	Executed	Completed, v1 (%)	Completed, v2 (%; 90% interval)
Claude API (GPT-6.1 Sol)	check-mtd-obligation	9	9	5(55.6)	5(55.6; 29.7to 79.1)
	create-send-invoice	5	5	0(0.0)	1(20.0; 3.6to 56.3)
	estimate-sole-trader-tax	2	2	1(50.0)	1(50.0; 9.7to 90.3)
	log-business-mileage	6	6	5(83.3)	5(83.3; 50.5to 97.0)
	mtd-deadlines	7	7	5(71.4)	6(85.7; 56.0to 97.4)
	prepare-quarterly-update	14	14	5(35.7)	5(35.7; 17.9to 57.5)
	record-transaction	14	14	8(57.1)	10(71.4; 49.7to 87.2)
	<i>All intents</i>	57	57	29(50.9)	33(57.9; 47.0to 68.2)
ChatGPT API (Claude Sonnet 5.5)	check-mtd-obligation	16	16	10(62.5)	12(75.0; 54.9to 88.9)
	create-send-invoice	6	6	0(0.0)	1(16.7; 3.0to 49.5)
	estimate-sole-trader-tax	1	1	0(0.0)	0(0.0; 0.0to 77.1)
	log-business-mileage	3	3	1(33.3)	1(33.3; 6.2to 76.4)
	mtd-deadlines	7	7	0(0.0)	0(0.0; 0.0to 23.2)
	prepare-quarterly-update	11	11	4(36.4)	3(27.3; 10.6to 51.8)
	record-transaction	10	10	5(50.0)	6(60.0; 34.7to 81.5)
	<i>All intents</i>	54	54	20(37.0)	23(42.6; 32.0to 53.8)

58 prompts can detect paired effects of about 10.3 points on the Claude API and 8.3 on the ChatGPT API with 80% power at the 5% level (Section 7.8). A change to `create_invoice` could only show in execution runs, which this experiment did not include.

The Lever is the method’s prediction of where points are available, so this experiment is its first test (falsification criterion F7). Against the median of TapTax’s own intents, the Lever placed the available points differently on the two hosts. On the Claude API it found 28.3 points at Selected for invoices and 12.0 for tax codes, and 40.7 at Surfaced for tax estimates. On the ChatGPT API almost everything sat at Surfaced: 49.1 points for tax estimates, 48.6 for tax codes and 13.5 for invoices, because the host rarely surfaced TapTax for the first two at all (posterior surfacing rates 5.7% and 2.6%). The realised per-intent changes were 0.0 (−12.5 to +12.5) for invoices and +8.3 (0.0 to +25.0) for tax codes on the Claude API, and −6.2 (−18.8 to 0.0) for tax estimates on the ChatGPT API, on 8, 6 and 8 prompts. The Lever is potential uplift, not a forecast of what a given rewording achieves, so a null intervention does not falsify it; but the one intervention tested realised none of the potential measurably, and per-intent intervals on so few prompts cannot tell. On the intents where the loss sits at Surfaced, rewording a tool can help only if the host searches, and on these intents the hosts mostly did not (Figure 4).

Server descriptions on the OpenAI path. The clearest evidence in the programme that metadata moves selection comes from a lever that exists on only one host. A measurement-only copy of the live TapTax version, identical except for a `server_description` that names its calculators, raised ChatGPT API selection on the calculator corpus’s pay prompts from 30.5% to 72.4%, a paired change of +42.0 points (+29.9 to +54.9), and on the property prompts from 2.7% to 44.0%, +41.3 (+26.0 to +56.7), pooling two replicates of each arm against three of the live version (Figure 5, right). On VAT prompts the change was +18.6 (+6.8 to +31.1), and on the MTD corpus +4.9 (−1.4 to +11.5), not distinguishable from zero. A second wording of the description reproduced the pay and property lifts. The lift had a price: on the property prompts TapTax calls on negative runs rose from 3 of 66 to 13 of 44 (4.5% to 29.5%, original labels). The same change cannot reach the Claude API, whose connector sends no server description. By contrast, every tool-description arm, on either host and either corpus, stayed within its noise (Table 9).

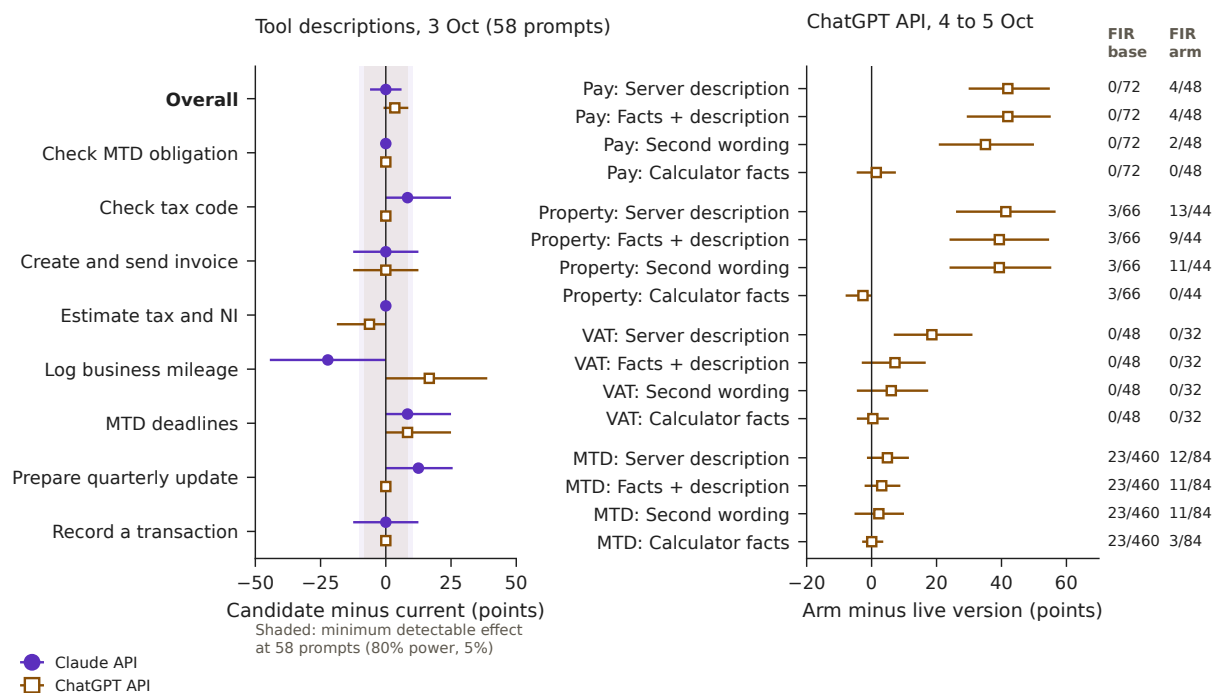


Figure 5: Metadata experiments, validation split. Left: the 3 October tool-description candidate minus the current version, overall and per intent (`taptax-val-current` and `taptax-val-candidate`; 58 positive prompts, 42 negative runs per arm); 90% prompt-cluster bootstrap; the shaded bands are each host’s minimum detectable effect at 58 prompts (80% power, 5% level) from the observed per-prompt SD. Right: ChatGPT API arms of 4 to 5 October against replicates of the live 1.1 version on the same prompts and pinned pool; 90% run-and-prompt bootstrap; text columns are TapTax calls over negative runs (original labels) in the live and arm suites. Evidence class S.

Tool inventory and versions. Version 1.1.0 added 33 calculator tools to the 15 of 1.0.0. On the Claude API its first paired comparison with 1.0.0 on the validation split showed -6.9 points (-12.1 to -2.6), and pooling all seven 1.0.0 runs on the same pool against both 1.1.0 runs gave -5.3 (run-and-prompt bootstrap -11.3 to 0.0). A later comparison on a changed competitor pool, six 1.0.0 runs against nine runs of the 1.1 live version, gave -0.7 (-5.2 to $+3.6$). On the ChatGPT API, all seven 1.0.0 runs against both 1.1.0 runs gave $+0.1$ (-3.3 to $+3.3$). A drop of a few points on the Claude API when the tool count tripled is plausible and not shown. The larger tool list also changed *which* TapTax tool was chosen: on the Claude API’s 1.1 live replicates, every first call on a tax-estimate prompt went to the new `calculate_tax_on_multiple_incomes` (21 runs) rather than `estimate_self_employed_tax` (0), and invoice requests began with `list_invoices` in 37 runs and `create_invoice` in 9. Selected counts all of these as TapTax’s; Section 8 returns to this construct gap.

7 Instrument validity

This section asks whether the instrument measures what it claims, with the TapTax data as the test bed.

7.1 Independent recomputation of the published path

Our Python port of the scoring path (Section 4.1) recomputed the pilot, baseline and same-day repeat reports for both surfaces from the per-run records. Every headline and intent estimate

Table 9: Metadata and tool-inventory experiments on the validation split. Baseline: the comparison arm (current version for the 3 October candidate; the 1.1.0 runs for the MTD arms of 4 October; replicates of the 1.1 live version for the 5 October arms). Change: arm minus baseline in points, paired over prompts, 90% interval from a prompt bootstrap (single runs) or a run-and-prompt bootstrap (replicated arms). FIR: TapTax calls over negative runs, original labels, baseline to arm. Runs: baseline and arm suites pooled. Hosts as in Table 3. Evidence class S.

Arm	Host	Prompts	Baseline (%)	Arm (%)	Change (90% interval)	FIR	Runs
3 Oct description candidate	Claude API	58	40.5	40.5	0.0(−6.0to +6.0)	0/42to 0/42	no
3 Oct description candidate	ChatGPT API	58	31.0	34.5	+3.4(−0.9to +8.6)	0/42to 0/42	no
MTD: Keep 15 tools	Claude API	58	35.3	36.2	+0.9(−6.5to +8.6)	0/84to 0/42	2 vs 1
MTD: Keep 23 tools	Claude API	58	35.3	40.5	+5.2(−0.9to +11.2)	0/84to 0/42	2 vs 1
MTD: Split server	Claude API	58	35.3	37.9	+2.6(−3.4to +8.6)	0/84to 0/42	2 vs 1
MTD: Split server, tax name	Claude API	58	35.3	36.2	+0.9(−4.3to +6.0)	0/84to 0/42	2 vs 1
MTD: Account wording	Claude API	58	35.3	40.5	+5.2(−1.7to +12.1)	0/84to 0/42	2 vs 1
MTD: Account wording	ChatGPT API	58	34.9	38.8	+3.9(−0.9to +9.9)	4/84to 4/84	2 vs 2
MTD: Disambiguation (1.1.1)	Claude API	58	35.3	33.6	−1.7(−6.9to +3.4)	0/84to 0/42	2 vs 1
MTD: Server description	ChatGPT API	57	35.5	40.4	+4.9(−1.4to +11.5)	23/460to 12/84	11 vs 2
MTD: Second description wording	ChatGPT API	57	35.5	37.7	+2.2(−5.3to +10.0)	23/460to 11/84	11 vs 2
MTD: Calculator facts	Claude API	58	39.3	42.7	+3.4(−1.0to +8.0)	0/377to 0/84	9 vs 2
MTD: Calculator facts	ChatGPT API	57	35.5	35.5	0.0(−2.9to +3.6)	23/460to 3/84	11 vs 2
MTD: Facts and description	ChatGPT API	57	35.5	38.6	+3.1(−2.2to +8.9)	23/460to 11/84	11 vs 2
MTD: Misroute fix	Claude API	58	39.3	40.1	+0.8(−3.4to +5.2)	0/377to 0/84	9 vs 2
MTD: Mileage wording	ChatGPT API	57	35.5	37.3	+1.8(−1.4to +5.1)	23/460to 2/84	11 vs 2
Pay: Server description	ChatGPT API	29	30.5	72.4	+42.0(+29.9to +54.9)	0/72to 4/48	3 vs 2
Pay: Second description wording	ChatGPT API	29	30.5	65.5	+35.1(+20.7to +50.0)	0/72to 2/48	3 vs 2
Pay: Calculator facts	Claude API	29	48.3	51.7	+3.4(−5.2to +13.2)	0/72to 0/48	3 vs 2
Pay: Calculator facts	ChatGPT API	29	30.5	31.9	+1.4(−4.6to +7.5)	0/72to 0/48	3 vs 2
Pay: Facts and description	ChatGPT API	29	30.5	72.4	+42.0(+29.3to +55.2)	0/72to 4/48	3 vs 2
Pay: Misroute fix	Claude API	29	48.3	47.4	−0.9(−7.8to +5.2)	0/72to 0/48	3 vs 2
Property: Server description	ChatGPT API	25	2.7	44.0	+41.3(+26.0to +56.7)	3/66to 13/44	3 vs 2
Property: Second description wording	ChatGPT API	25	2.7	42.0	+39.3(+24.0to +55.3)	3/66to 11/44	3 vs 2
Property: Calculator facts	Claude API	25	34.0	32.0	−2.0(−11.3to +7.3)	0/66to 0/44	3 vs 2
Property: Calculator facts	ChatGPT API	25	2.7	0.0	−2.7(−8.0to 0.0)	3/66to 0/44	3 vs 2
Property: Facts and description	ChatGPT API	25	2.7	42.0	+39.3(+24.0to +54.7)	3/66to 9/44	3 vs 2
VAT: Server description	ChatGPT API	22	18.9	37.5	+18.6(+6.8to +31.1)	0/48to 0/32	3 vs 2
VAT: Second description wording	ChatGPT API	22	18.9	25.0	+6.1(−4.5to +17.4)	0/48to 0/32	3 vs 2
VAT: Calculator facts	Claude API	22	23.5	28.4	+4.9(0.0to +12.1)	0/48to 0/32	3 vs 2
VAT: Calculator facts	ChatGPT API	22	18.9	19.3	+0.4(−4.5to +5.3)	0/48to 0/32	3 vs 2
VAT: Facts and description	ChatGPT API	22	18.9	26.1	+7.2(−3.0to +16.7)	0/48to 0/32	3 vs 2
VAT: Misroute fix	Claude API	22	23.5	26.1	+2.7(−3.0to +9.5)	0/48to 0/32	3 vs 2

and both interval bounds agree with the production reports, with a largest absolute difference below 10^{-13} points; because the random number generator is ported exactly, the Monte Carlo intervals agree draw for draw, not only in distribution. The arithmetic of the published path is therefore reproducible from the data pack.

The recomputation also found six places where earlier written summaries of the programme, including the brief this paper was written from, disagree with the data.

1. **The pilot’s intra-prompt correlations.** The PRD reports pooled pilot ρ of 0.71 (Claude API) and 0.51 (ChatGPT API) at Surfaced and 0.85 and 0.78 at Selected given Surfaced (Amos, 2026, section 36). The same estimator on the pilot runs in the pack gives 0.90, 0.89, 0.89 and 0.89. We could not find a filter or estimator that reproduces the reported values. The code’s default $\rho_D = 0.6$ and $\rho_S = 0.8$ were chosen from the reported values.
2. **Two definitions of a false invoke.** The “11 of 132” figure for the ChatGPT API counts any external call on a negative prompt; the published False-Invoke Rate counts TapTax calls only, 5 of 132 (Section 6.5).
3. **The decoy’s floors.** The decoy check can rule out decoy-selection rates above 3.2% on the 58 positive prompts and above 8.6% on the 21 negative prompts (90% Jeffreys upper bounds on zero of n), not about 8.6% on positives as the brief stated.
4. **The report’s ρ field.** The integration-level ρ printed in each report (for example $\rho_D = 0.91$ and $\rho_S = 0.00$ on the Claude API baseline) is the first intent cell’s estimate, `check-mtd-obligation`, not a pooled value.
5. **A replicated version comparison.** A working-session figure of -3.0 points (-8.6 to $+2.2$) for 1.0.0 against 1.1.0 after replication could not be reproduced from any combination of suites in the pack; the recomputed comparisons are in Section 6.7.
6. **The degenerate resolution split** of Section 6.3.

None of these changes a published headline. All of them change what a reader of the internal record would believe, which is why every number in this paper is recomputed and carries its provenance.

7.2 Instrument controls

The control round of 4 October 2026 (round 2026-10-04) ran four controls on the validation split of the baseline suite (58 positive and 21 negative prompts, $K = 2$, 5 rotations) on both surfaces (Figure 6, Table 10). On the ChatGPT API the round finished on 5 October after the OpenAI account ran out of credit and 247 runs were re-sent; we use the complete round, and the earlier partial evaluation is superseded.

Placebo. This is an A/A test (Kohavi et al., 2020). Two arms identical to the base suite differed by -2.6 points on the Claude API (95% interval -6.9 to $+1.7$) and $+2.6$ on the ChatGPT API (-1.7 to $+6.9$). Both intervals cover zero, so the control holds; at the 95% level a sound instrument still fails about one round in twenty.

Blinded metadata. Replacing TapTax’s tool names, titles, descriptions, schema text and server text with neutral placeholders cut selection from 41.4% to 12.9% on the Claude API (-28.4 points, -39.7 to -18.1) and from 31.0% to 7.8% on the ChatGPT API (-23.3 , -33.6 to -13.8). The runs read the metadata an integration controls, which is the precondition for any metadata experiment to mean anything.

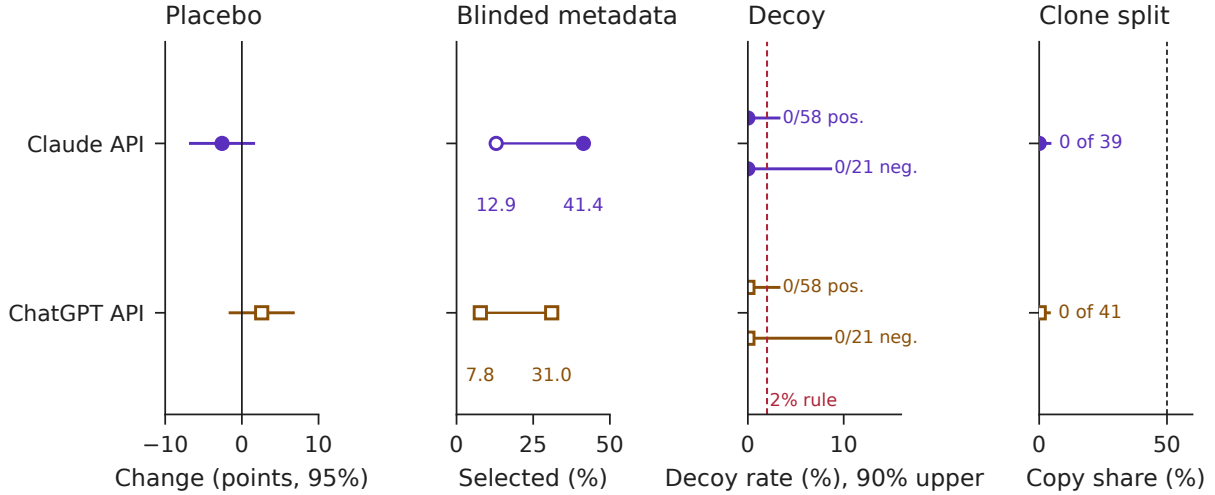


Figure 6: Instrument controls, round 2026-10-04, validation split, both hosts, recomputed from `runs.csv` (suites `taptax-uk-ctl-2026-10-04-*`) and agreeing with `checks.csv`. Placebo: placebo minus reference, paired over prompts, 95% prompt-cluster bootstrap. Blinded: selection with full metadata (filled) and with tool names, titles, descriptions and schema text removed (open). Decoy: zero picks with the 90% Jeffreys upper bound on positive and negative prompts, against the 2% rule. Clone split (retired design): the share of first picks won by the copy labelled `taptax-2`, 90% Jeffreys interval, against one half. Evidence class S.

Table 10: Instrument-control verdicts, round 2026-10-04 (complete round). Runs and cost as recorded in `checks.csv`; the placebo row counts the shared reference arm.

Host	Check	Rule	Statistic	Interval	Verdict	Runs	Cost (\$)
Claude API	Placebo	95% interval of Δ cov-ers 0	$\Delta -2.6$	-6.9to +1.7	holds	316	3.98
	Blinded metadata	falls by half; 95% upper bound < 0	41.4to 12.9; $\Delta -28.4$	-39.7to -18.1	holds	158	1.89
	Decoy	fails only if lower bound > 2%	0of 158runs	upper 3.2(pos.), 8.6(neg.)	holds	158	1.99
	Clone split (retired)	copy's share covers 1/2	copy 0of 39picks	0.0to 4.8	fails	158	1.98
ChatGPT API	Placebo	95% interval of Δ cov-ers 0	$\Delta +2.6$	-1.7to +6.9	holds	316	2.30
	Blinded metadata	falls by half; 95% upper bound < 0	31.0to 7.8; $\Delta -23.3$	-33.6to -13.8	holds	158	1.20
	Decoy	fails only if lower bound > 2%	0of 158runs	upper 3.2(pos.), 8.6(neg.)	holds	158	1.16
	Clone split (retired)	copy's share covers 1/2	copy 0of 41picks	0.0to 4.5	fails	158	1.14

Decoy. An unrelated public weather server added to the pool was never chosen (the name is a specificity check, unrelated to the decoy effect of choice theory (Huber et al., 1982)): 0 of 158 runs on each surface, and it was never even surfaced on either host. The rule passes because it fails only when the lower bound exceeds 2%. The sample cannot vouch for a floor near 2%: its 90% upper bounds are 3.2% on positive prompts and 8.6% on negatives. The decoy is also far from the pool’s subject, so it tests gross failures of the plumbing, not the resolution between near-competitors.

Clone split. An exact copy of TapTax was added to the pool under the label `taptax-2`, beside the original `taptax`. On the Claude API the copy won 0 of 39 first picks (90% upper bound 4.8%), and on the ChatGPT API 0 of 41 (4.5%), against an expected one half. Raw events show that on the Claude API both copies surfaced side by side in 80 positive runs and their references resolved correctly; the copy was called only as a fallback after the replay server refused the first call. Listing order does not explain it. Faghih et al. (2025) found that models given two identical tools pick the first-listed one most of the time; here the copy lost whether

it was listed first (0 of 15 picks on the Claude API, 0 of 18 on the ChatGPT API) or second (0 of 24, 0 of 23). In this design the label and the server’s identity were confounded, so the failure shows that the hosts decide between otherwise identical servers by their names, not that the plumbing treats identical servers differently; it is consistent with evidence that the match between a request and a tool’s metadata, its name included, is the strongest driver of choice among equivalent tools (Blankenstein et al., 2026). It is the central piece of evidence for RQ3: a control that a naive design would expect to pass exposed a label sensitivity strong enough to decide every pick. The redesigned control, `clone_swap`, gives both copies matched opaque labels (`srv-` plus four characters) and crosses labels and listing order in a balanced 2×2 , reporting pair share, label bias and position bias separately (Section G). It is built and has not been run.

7.3 Run-to-run variance

Hosted models are not deterministic even at temperature zero, through batch-dependent numerics among other causes (He and Thinking Machines Lab, 2025; Yuan et al., 2025; Atıl et al., 2025), and repeated runs of agentic evaluations disagree by more than single-run error suggests (Bjarnason et al., 2026). Earlier working summaries described identical TapTax servers scoring “anywhere from 36 to 44%” on the Claude API and inferred a session-level variance component that prompt-level intervals hide. The data pack allows a direct test. We grouped every replicate of an identical arm: the same pinned TapTax version, the same pinned competitor versions, the same corpus, split, K and rotations, differing only in when and how it was run. Two pools must be kept apart, because one competitor’s pinned version changed between 4 and 5 October. On the Claude API this gives 7 runs of TapTax 1.0.0 on the first pool, 6 on the second and 9 runs of the 1.1 live version; on the ChatGPT API, 7 runs of 1.0.0 and 11 of the live version (Figure 7).

With the prompts fixed, a run’s selection rate has a within-run sampling variance $\sum_i p_i(1 - p_i)/N^2$ over its N trials, which we estimate from the pooled per-prompt rates. If runs are exchangeable at the trial level within each prompt, the observed variance of the run rates should match it; a within-prompt permutation of outcomes across runs gives an exact test.

On the Claude API there is no excess. The observed SD of run rates was 1.7, 1.3 and 1.5 points in the three groups, against within-run SDs of 2.2, 2.2 and 2.2; the permutation p -values were 0.776, 0.900 and 0.879. The spread of identical Claude API runs is what binomial sampling of 116 trials per run implies, and the method-of-moments between-run SD is 0.0. The 36 to 44 range came partly from a measurement-only variant (`keep15`) that is a different version record from 1.0.0; paired against the 1.0.0 run it was compared with, it differed by -7.8 points (-14.7 to -1.7), an interval that excludes zero. The pack cannot show whether the two servers were byte-identical, so this pair is either a real difference between the two records or a chance exceedance; it is not evidence of run-level variance.

On the ChatGPT API the first pool shows an excess. The 7 runs of 1.0.0, spread over 26.9 hours on 3 and 4 October, had an observed SD of 3.3 points against 2.0 within runs (variance ratio 2.53, permutation $p = 0.014$), a between-run SD of about 2.5 points. The 11 live-version runs, started within 8.8 hours, did not (ratio 1.26, $p = 0.254$). The ChatGPT API excess is therefore plausibly temporal drift across days rather than per-session noise, and two pools are not enough to tell.

Two consequences follow for the method. First, its response to the earlier summaries, replicating any comparison whose first pass excludes zero and pooling arms with a bootstrap that resamples runs as well as prompts, is sound as protection on the ChatGPT API but costs power on the Claude API, where it is not needed; the PRD’s rule “never resample runs” (Amos, 2026, section 17.2) now disagrees with the code. Second, the apparent instability of single comparisons on the Claude API is the multiple-comparisons problem, not a missing variance component: a programme that runs dozens of paired comparisons at 90% will see several intervals exclude zero by chance. The same-day repeat’s paired change of $+2.6$ points on the Claude API (Section 6.6) is one such interval, and it coincides with a change of execution path.

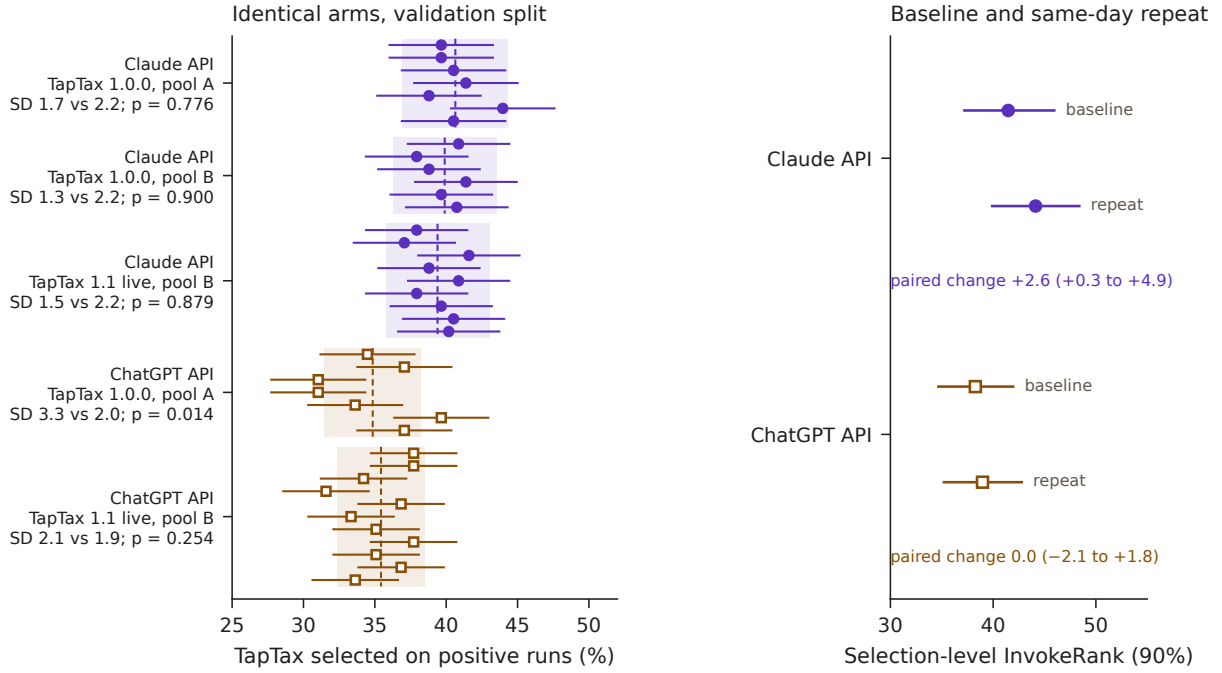


Figure 7: Run-to-run variance. Left: each replicate of an identical arm (validation split, positive runs, prompts common to every run of its group; suites listed in the figure’s data table), with its within-run 90% range given the prompts (± 1.645 times the within-run SD); the shaded band is the group mean plus or minus the same amount, the spread expected from within-run sampling alone; labels give the observed SD of run rates against the within-run SD and the within-prompt permutation p -value (4,000 permutations). Right: baseline and same-day repeat with published 90% credible intervals, and the paired change with its 90% prompt bootstrap. Evidence class S.

7.4 Repeat clustering

Repeats of a prompt are strongly correlated on both hosts (Figure 8). Pooled over the baseline’s prompts, ρ at Surfaced was 0.94 on the Claude API and 0.91 on the ChatGPT API, and at Selected given Surfaced 0.81 and 0.87. These pooled values include between-intent differences, because every prompt of every intent is one group; removing intent means gives 0.90 and 0.74 at Surfaced and 0.76 and 0.86 at Selected. At these values a second repeat adds little: at the pooled ρ_S a prompt run twice carries the information of about 1.1 independent runs on the Claude API and 1.1 on the ChatGPT API.

The per-cell estimates the published path actually uses are unstable. Of the 15 Claude API cells where a cell could estimate its own ρ at Surfaced or Selected, 4 sat at 1 and 1 at 0; on the ChatGPT API 3 of 12 sat at 1 and 1 at 0. A cell at 1 counts each prompt once, which is conservative; a cell at 0 applies no deflation and overstates the information in its runs by up to a factor of K . The cell `check-mtd-obligation` on the Claude API has $\rho_S = 0$, so its Selected interval is as narrow as if its 44 runs were independent. Pooled or within-intent ρ with shrinkage toward the pooled value would remove this instability at no cost to the point estimates.

7.5 Interval calibration

Alternative intervals on the baseline. A prompt-cluster bootstrap of the plug-in estimate (equal-weight mean over intents of the selected share, resampling prompts within intents) gives 41.3 (36.6 to 46.1) on the Claude API and 38.6 (33.9 to 43.2) on the ChatGPT API. The published intervals are 94.2% and 81.2% as wide as the bootstrap’s. On the ChatGPT API the model interval is therefore 18.8% narrower than an interval that respects the prompt clustering

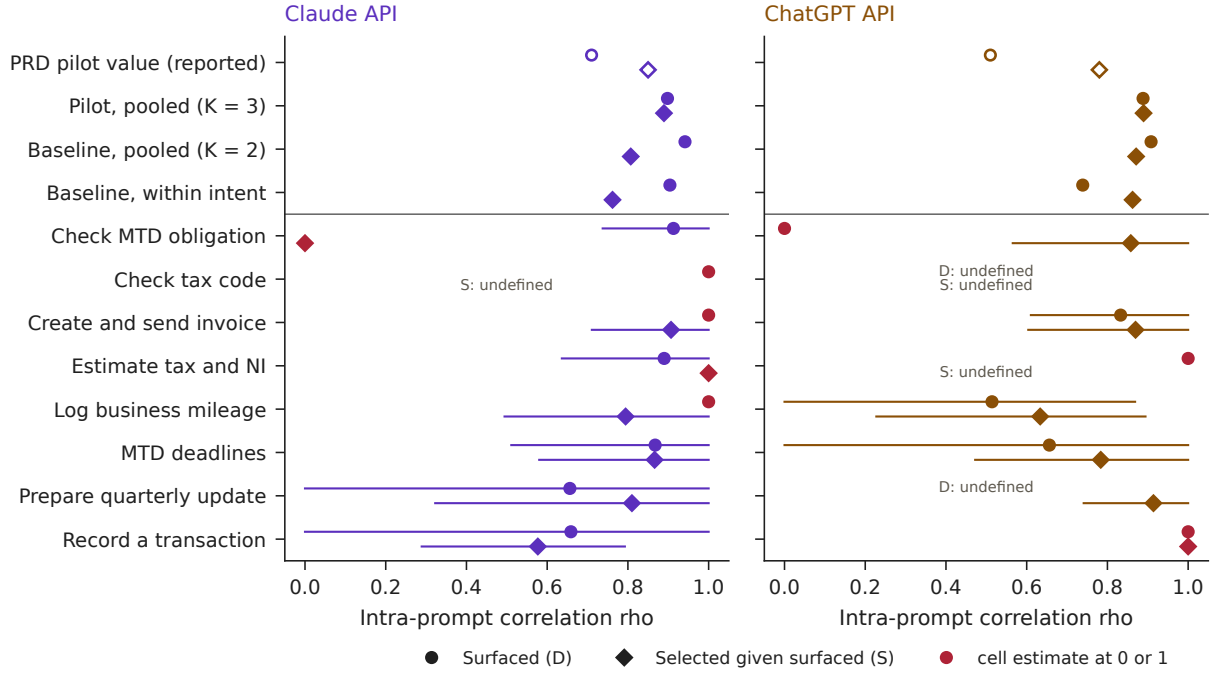


Figure 8: Intra-prompt correlation ρ by stage and host: the PRD’s reported pilot values (open), the pooled pilot ($K = 3$) and baseline ($K = 2$) estimates as the published fallback computes them, the within-intent baseline estimate, and every per-cell baseline estimate with a 90% prompt bootstrap interval; red markers are cell estimates truncated at 0 or 1, which the published path then uses as they are. Fleiss ANOVA estimator. Source: `runs.csv` (taptax-uk, pilot and measurement). Evidence class S.

without a model; Section H shows that most of the difference comes from partial pooling across intents rather than from the ρ estimates. The prior cap did not bind on the baseline: plain empirical Bayes gives the same intervals (41.5, 37.1 to 46.1 on the Claude API).

Dependence between stages. Across prompts, the share of a prompt’s runs that surfaced TapTax and the share of its surfaced runs that selected it were nearly uncorrelated: Pearson $r = 0.09$ (90% bootstrap -0.08 to 0.22) over 135 prompts on the Claude API and 0.10 (-0.05 to 0.24) on the ChatGPT API. A model-based estimate of the latent cross-stage correlation is larger (Section H), but the simulation that uses it finds that independent stage draws are not what limits coverage; the narrower ChatGPT API interval comes mostly from partial pooling across intents, which holds even when every prompt is counted once (Section H).

Coverage by simulation. A simulation calibrated to the TapTax baseline (Section H, Figure 10) calls the scoring package itself and compares it with alternatives. At the calibrated centre the published 90% roll-up intervals covered 87.1% of the time on the Claude-calibrated design and 82.2% on the ChatGPT-calibrated design, and per-intent intervals 84.2% and 82.5%: modest undercoverage. Two conditions make it worse. Under the scoring test’s own design of three identical intents, per-intent coverage was 87.5%, as the test claims, but roll-up coverage was 65.0%, because the cap makes each intent’s posterior nearly the pooled one and the roll-up then treats them as independent. And with a between-run SD at the value fitted to the ChatGPT API’s first replicate pool, roll-up coverage fell to 74.7%. A generalised linear mixed model with prompt random effects stayed near nominal (89.0% and 90.4% at the centre), except where a run component it does not model was present; a bootstrap over runs and prompts with two sessions per study restored nominal roll-up coverage there.

Table 11: Unique paired prompts for 80% power to detect a paired change of δ points. PRD: $\sigma_d = 0.35$. Observed: σ_d of the 3 October metadata pair on each host (0.28 and 0.23). With runs: the ChatGPT API’s between-run SD (2.5 points), with one or three runs per arm; “none” means no number of prompts suffices. Per-prompt differences average the $K = 2$ repeats, so the prompt-level clustering is already in σ_d .

δ	α	PRD	Observed, Claude API	Observed, ChatGPT API	ChatGPT API, 1 run per arm	ChatGPT API, 3 runs per arm
3	0.05	1,069	689	449	none	none
3	0.10	842	543	354	none	none
5	0.05	385	248	162	none	none
5	0.10	303	196	128	none	none
10	0.05	97	62	41	none	61
10	0.10	76	49	32	155	44

7.6 Judge calibration

Completed rests on an uncalibrated cross-family judge (Section 6.4). Three threats compound. The rubric was revised after inspecting verdicts, which moved 6 and 5 runs to completed and 2 and 2 the other way. Judge family and host are confounded, so a host difference in completion may be a judge difference. And the planned gold-set labeller is the owner of both systems. An independent second annotator with inter-annotator κ , both judges scoring both hosts, and prediction-powered inference on the gold set are the minimum before a completion figure can support a claim (Angelopoulos et al., 2023; Zheng et al., 2023; Panickssery et al., 2024).

7.7 Eligibility fairness

No neutral eligibility exists, so no competitive figure in this paper is valid as a comparison between products. The baseline’s competitors were scored on TapTax-authored intents, which by construction suit TapTax; their low selection rates (a competitor was the first call in 4 and 18 of 390 positive runs) say little about them. Section F specifies the neutral process and the sensitivity analysis that should accompany it.

7.8 Power and sample size

For paired designs the planning quantity is the spread of per-prompt differences, σ_d . The PRD plans with $\sigma_d = 0.35$, for which about 96 prompts detect a 10-point paired change with 80% power at the 5% level, and 58 prompts detect about 12.9 points. The observed σ_d of the 3 October metadata pair was smaller, 0.28 on the Claude API and 0.23 on the ChatGPT API, and that of the placebo pair 0.17, so the PRD’s rule is conservative: at 58 prompts the observed minimum detectable effects are 10.3 and 8.3 points (Figure 9). Detecting 5 points at the 5% level needs about 248 unique paired prompts on the Claude API and 162 on the ChatGPT API; 3 points needs 689 and 449. The validation split cannot resolve changes as small as the +3.4 points the metadata experiment observed on the ChatGPT API.

A between-run component changes the arithmetic. If each arm is one run and runs carry an independent SD σ_{run} , the variance of a paired difference gains $2\sigma_{\text{run}}^2$, which no number of prompts removes. With the ChatGPT API first pool’s 2.5 points, a single-run paired comparison cannot detect less than 10.0 points however many prompts it uses; three runs per arm bring 10 points within reach of 61 prompts. On the Claude API, where no between-run component was detected, prompts are the binding constraint. Table 11 gives the planning numbers.

The method’s anytime-valid alternative, stopping on discordant prompts once a beta-mixture likelihood ratio reaches 20 after at least 40 prompts, saves runs when an effect is large and does not change these planning numbers when it is small.

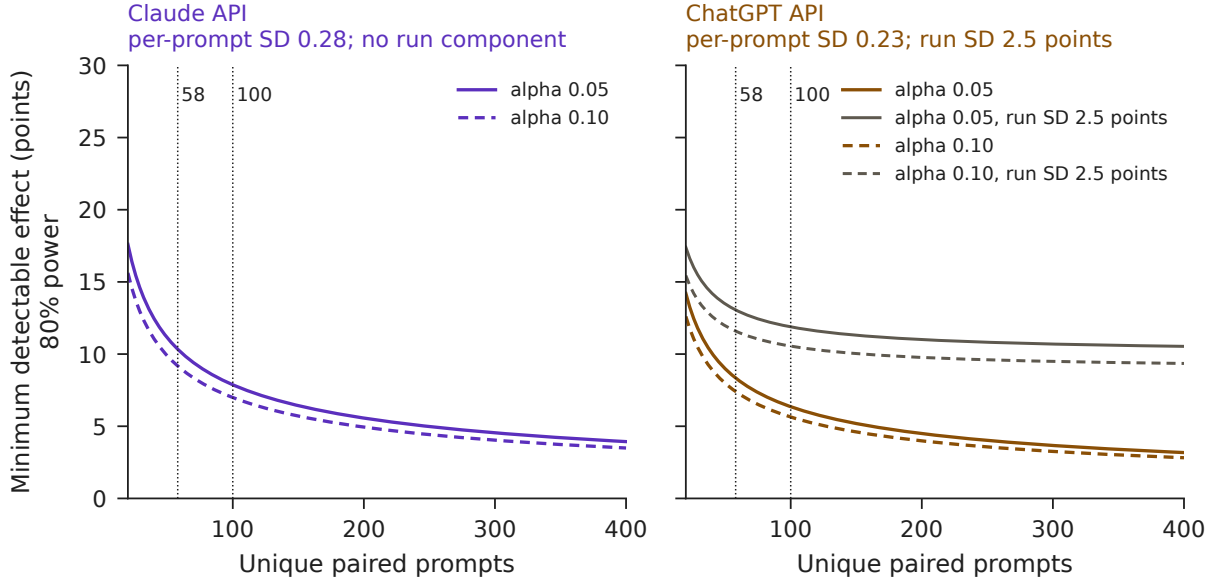


Figure 9: Minimum detectable paired effect (80% power) against unique paired prompts, from the observed per-prompt SD of the 3 October metadata pair; grey: with the between-run SD estimated from the ChatGPT API’s first replicate pool, one run per arm. Vertical lines: the validation split (58 prompts) and the PRD’s planning default (100). Derived from `runs.csv` (`taptax-val-current`, `taptax-val-candidate` and the replicate groups of Figure 7).

Table 12: External validity (RQ8): real TapTax tool calls by host in the 90 days to 5 October 2026 (T evidence, test accounts excluded) against the minimum the comparison needs, and human-assisted (A) studies. Source: `telemetry.csv` and `checks.csv` (kind `real_usage`, `checks-v2`).

Host	Real calls	Needed	Window at current rate	A studies	Verdict
Claude	0	30	no rate to project	none	insufficient
ChatGPT	8	30	about 338 days	none	insufficient

7.9 External validity

The method compares the synthetic tool mix with an integration’s real calls on the same host, and separately with human-assisted runs in the consumer apps, and never pools the three (InvokeRank, 2026c). Neither comparison can be made for TapTax (Table 12). External validity to real consumer behaviour remains unresolved because production volume is insufficient and no human-assisted study has been recorded; we compute no synthetic-to-production correlation (RQ8).

7.10 Falsification criteria

Table 13 states how the method could be shown wrong and where each test stands.

8 Threats to validity and limitations

Construct: eligibility. The baseline’s positive prompts come from intents written with knowledge of TapTax’s capabilities, so “eligible” means “TapTax-shaped”. Every competitive quantity therefore measures a TapTax-authored category, which favours TapTax by construction. The neutral construction of Section F addresses this, and it has its own threats: incomplete competitor discovery (the index is the sampling frame), ambiguous capability boundaries, unequal

Table 13: Falsification criteria and their status on 6 October 2026.

Criterion	Test	Status
F1 Funnel incoherence: end-to-end success disagrees with the product of measured stages	Holds by construction for nested events; testable only as recording error, by comparing execution trajectories with stored events	No incoherence: no run reached a stage without its parent
F2 Interval undercoverage	Simulation calibrated to TapTax (Section 7.5); later, repeated studies against a known truth	Modest undercoverage at the TapTax calibration (82.2% to 87.1% for nominal 90%); severe for the roll-up when intents are alike (65.0%) or a run component is present
F3 Persistent clone asymmetry after opaque labels and balanced rotation	<code>clone_swap</code> round on both hosts	Naive design failed on both hosts; balanced design not run
F4 Placebo (A/A) inflation	Placebo rounds; replicated identical arms	Placebo holds on both hosts; no excess run variance on the Claude API; excess on one ChatGPT API pool ($p = 0.014$)
F5 External-validity failure	S against T, and S against A, per host	Untestable: 0 and 8 real calls in 90 days, no A study
F6 Eligibility instability	Rank stability across alternative eligibility matrices	Untestable: no neutral eligibility
F7 Lever failure	Interventions against predicted Lever	One null intervention; per-intent intervals too wide to test
F8 Judge invalidity	Gold set, κ , TPR and TNR per judge	Untestable: 0 gold labels
F9 Temporal instability beyond modelled uncertainty	Week-apart repeat; replicate pools over days	Same-day repeat overlaps but its Claude API paired change excludes 0; week-apart repeat not run

documentation (eligibility is judged from descriptions, which advantages well-documented tools), region and account restrictions, read versus write capability, version drift between the snapshot that was judged and the one that was run, annotator disagreement between two model families and one human reviewer who also owns the focal integration, and the treatment of *unknown*. Reporting rank stability across alternative eligibility matrices (unknown as eligible, unknown as ineligible, each family alone) is the minimum sensitivity analysis.

Construct: Selected is integration-level. A selection counts for TapTax whichever of its tools is called. On a 48-tool server this hides wrong-tool picks: with the 1.1 tool list, every Claude API first call on a tax-estimate prompt went to a multiple-incomes calculator rather than the self-employed estimate, and most invoice requests began by listing invoices (Section 6.7). Whether those picks are wrong depends on the prompt; the working sessions classified them with fit rules partly authored with the TapTax side and partly decided by blind two-family prompt labels, which are not in the data pack. Intra-integration tool correctness is a construct gap between Selected and Completed that the funnel does not name.

Construct: the corpus. Of the 320 prompts, 37 depend on context the harness never supplies and remain in every figure until the human review is done. Such prompts depress every integration’s score alike, but they also inflate the no-tool share, which is the paper’s largest finding.

Internal: judge and rubric. Completion rests on an uncalibrated judge, a rubric revised after inspection, a judge family confounded with host, and a planned labeller who owns both systems (Section 7.6).

Internal: researcher degrees of freedom. The programme ran 136 suites in 21 families over three days, several of them chosen after seeing earlier results (Table 5). Only the 3 October

metadata experiment and the control round were specified before their data existed. The server-description finding is large and replicated within its session, but it came from an exploratory arm and needs confirmation on fresh prompts, ideally on the locked test split under a pre-registered analysis.

Statistical. Per-cell ρ is unstable at $K = 2$ (Section 7.4); the model interval is narrower than a prompt-cluster bootstrap on the ChatGPT API (Section 7.5); the prior cap’s coverage depends on the data-generating process (Section 7.5); per-intent claims are many and individually underpowered. The implementation uses Benjamini-Hochberg at $q = 0.10$ for per-intent claims (e-BH when sequential), a single pre-registered pooled primary test, and lists rather than flags per-intent intervals that exclude zero (Benjamini and Hochberg, 1995; Wang and Ramdas, 2022); this paper follows the same discipline and reports per-intent intervals as descriptive.

External: API versus consumer application. API runners differ from consumer apps in system instructions, hidden retrieval, entitlement, personalisation, memory, model routing, tool ranking, safety policy, post-processing and interface state. The claude.ai system prompt does not apply to the API, and ChatGPT tool invocation depends on plan, workspace, role, surface and region. The finding that server **instructions** reach neither API path is an example: they may matter in the consumer apps. In general $P_{\text{API}}(S \mid D) \neq P_{\text{consumer}}(S \mid D)$. External validity to real consumer behaviour remains unresolved (Section 7.9).

External: one integration, one category, two models. Everything here is one owned integration in one UK tax category, measured on one model per host over three days. Model updates, other categories and other hosts are untested.

Conflict of interest. One person owns TapTax and InvokeRank, approves competitor sets, and is the planned gold-set labeller, eligibility reviewer and corpus reviewer; the candidate descriptions and some tool-fit rules were written with an AI agent working on TapTax’s side. The owned system is useful because it gives ground truth at every stage; it is insufficient for generalisation, and the next validation domain should be an external design partner in a different category.

Ethical considerations. A public ranking of how agents distribute software creates an incentive to optimise for the ranking. The method’s guardrail lints every candidate description for superlatives, imperatives aimed at the model, references to other tools, hidden or instruction-like text and over-broad triggers, and requires that false invokes do not rise (Amos, 2026, section 17.4); the guardrail’s own validity is untested, and the server-description arm shows that a truthful description can still raise false invokes. Tool metadata is also a prompt-injection channel, so measurement suites must treat competitors’ metadata as untrusted input, which the replay server does by serving it verbatim and never executing it. Measurement error has economic consequences when rankings are public: an interval that is too narrow can turn a tie into a ranking, which is one reason the coverage results in Section 7.5 matter beyond statistics. Telemetry is metadata-only by default (tool, host, outcome, duration, hour), with no user identifiers and with test accounts excluded.

What InvokeRank does not establish. Under the tested conditions InvokeRank measures the probability of surfacing, selection, valid invocation, execution and completion; the conditional distribution among eligible integrations; native versus external resolution; and host-specific behaviour. It does not by itself establish market or revenue share, user satisfaction or welfare, causal discrimination by a host, a platform’s internal ranking algorithm, behaviour after

model changes, security, the correctness of a vendor’s domain logic, or consumer-app behaviour from API results.

9 Outstanding validation

On 6 October 2026 none of the following has been done; each was checked against the data pack exported that day.

1. The neutral category build and reviewed eligibility (Section F).
2. Native self-service judging and the Agent Resolution Split: no native verdicts exist.
3. The human review of the 91-prompt corpus realism sample and the corpus revision.
4. The 300-run gold set, κ , the judge’s true and false positive rates and prediction-powered completion.
5. The `clone_swap` round on both hosts.
6. The week-apart repeat, scheduled for 10 October 2026 subject to the owner’s approval of its spend.
7. A human-assisted consumer-app study on either host.
8. Enough real telemetry for an S-against-T comparison.
9. Any run on the locked test split.
10. A second, independent category or design partner.
11. A full simulation study of interval coverage beyond the one reported in Section 7.5, including several runs per study.

We recommend a pre-registered confirmatory analysis on the locked test split (45 positive and 14 negative prompts) once the corpus review and the neutral eligibility exist. The pre-registration should fix the primary estimand (Selection-level InvokeRank per host), the server-description hypothesis on the OpenAI path with its false-invoke guardrail, the interval method (a prompt-and-run bootstrap with at least three runs per arm on the ChatGPT API), and the multiplicity rule for per-intent claims.

10 Discussion

The no-tool branch is where distribution is decided. The largest share of eligible requests never reached any integration, and competitors were rarely the alternative. Measured as a tool-use benchmark measures, conditional on a tool being used, TapTax would look dominant: on the Claude API it was the first call in 162 of the 162 plus 4 runs with an external call. Measured as distribution, it was chosen in 41.5% of positive requests. The two-level view of Equation (4) separates two decisions that interventions address differently. A host that does not search (the Claude API’s dominant no-tool mode) cannot be reached through tool metadata, because the metadata is never retrieved; a host that searches and declines (the ChatGPT API’s mode) has read the candidates and preferred its own answer. This links distribution measurement to the literature on when models should consult external sources at all (Mallen et al., 2023; Asai et al., 2024; Wang et al., 2023; Ross et al., 2025; Qian et al., 2025; Song et al., 2025), and it means the native self-service judge is not an optional extra: until it runs, a no-tool outcome may be a correct answer or a failure, and the paper cannot say which.

The levers differ by host. On the Claude API path the model sees tool names and descriptions only; on the OpenAI path it also sees a server description that the caller sets from the directory listing. The one large metadata effect in the programme came through that second channel, and it cannot transfer to the first. It also came with a false-invoke cost on one corpus, which is what the guardrail of comparing false-invoke rates exists to catch. This sits close to work showing that edits to tool descriptions can multiply a tool’s usage and that tool metadata is an attack surface (Faghih et al., 2025; Sneh et al., 2025; Wang et al., 2026c,b; Greshake et al., 2023; Aggarwal et al., 2024); the distinction that matters for a measurement method is between truthful optimisation (a description that tells the host what the tool actually does) and manipulation (one that induces calls the tool cannot serve). The server description in our arms was truthful, and it still raised false invokes on property prompts; a false-invoke rate on independently labelled negatives is the minimum evidence that a lift is not bought with indiscriminate calling.

Names decide between near-identical servers. The clone failure is the result that most limits competitive measurement. If two byte-identical servers split first picks 0 to 39 by label, then any comparison between competitors whose capabilities overlap measures their names as much as their tools. This is consistent with position and label biases documented for model choice among options (Zheng et al., 2024a; Pezeshkpour and Hruschka, 2024; Blankenstein et al., 2026), and it means competitive claims need the redesigned control’s resolution limit before they say that one integration beats another.

Execution is not completion. A server that answers every call can still leave most tasks unfinished, because the host stops after a first read, or does not relay the result. For vendors, this places part of distribution in the host’s orchestration, which tool metadata may or may not influence; for measurement, it means Executed is a weak proxy and a calibrated completion judge is required before any full InvokeRank is published. The judge’s own validity, including family effects, is an open problem in this setting (Zheng et al., 2023; Panickssery et al., 2024; Lù et al., 2025).

What the statistics need. Three changes would make the published intervals more defensible on the evidence here. First, replace per-cell ρ with a pooled or within-intent estimate shrunk toward it, because cells at $K = 2$ often land on 0 or 1. Second, use at least three runs per arm on the ChatGPT API, or model the run as a random effect, because a between-run component there sets a floor on detectable effects that more prompts cannot lower. Third, plan experiments on unique prompts from observed σ_d : the validation split’s 58 prompts resolve only effects of about 8.3 to 10.3 points. The simulation (Section 7.5) shows where the capped prior’s intervals hold and where they do not, and the prompt-cluster bootstrap is a cheap cross-check that the method could report beside every headline.

Relation to prior work and novelty. Most of the method’s statistics are re-applications: the chain rule, beta-binomial empirical Bayes, Jeffreys intervals, design effects, paired cluster bootstraps, pass^k , the conditional logit and its Elo-scale display, and prediction-powered inference. Loss share is the logarithmic index decomposition of a product (Ang, 2015), equal to a normalised Shapley attribution of the log loss, and Lever is the improvement potential of reliability engineering generalised to a reference (Birnbaum, 1968; Rausand et al., 2020); Invoke Share is a share of choice (Luce, 1959; Train, 2009). None is a contribution of this paper. Each element of the design also has precedent: staged views of tool use (Qu et al., 2025b; Li et al., 2023), the decision whether to use a tool at all (Huang et al., 2024; Ross et al., 2025; Song et al., 2025), retrieval as a separate failure stage in MCP settings (Mo et al., 2026; Shi et al., 2025), selection among functionally equivalent tools (Blankenstein et al., 2026), identical-tool and metadata-edit

experiments (Faghih et al., 2025), and replayed tool servers (Guo et al., 2024). What we could not find is their combination as a distribution measurement: every stage observed on the hosts’ own deferred tool-search pipelines, with the host’s own answer as an outside option, independent eligibility defining the competitive denominator, evidence classes that are never pooled, and a pre-specified battery of controls run on an integration with ground truth at every stage [UN-VERIFIED]. That combination, and the clone-label and server-description findings it produced, are the paper’s contribution.

11 Conclusion

The current evidence supports five claims, each under the tested conditions: one integration, one UK tax category, one model per host, API runners, 3 to 5 October 2026.

1. The nested funnel is measurable on real host tool-search pipelines, its product is an exact probability, and the published scoring path is arithmetically reproducible from per-run records.
2. For TapTax the dominant alternative to the integration was no external tool, not a competitor, and the two hosts reached that outcome differently.
3. The runs respond to the metadata an integration controls (blinded control), a null arm returns null (placebo), and an unrelated server is not chosen (decoy).
4. Hosts decide between byte-identical servers by their labels, so competitive comparisons need a label-and-position resolution limit before they can be trusted.
5. On the OpenAI path a truthful server description can move selection by tens of points on some intents, at a cost in false invokes; tool-description rewrites did not move selection measurably on either host.

Table 14 states every claim of the paper with its status.

The evidence does not yet support any claim about completion (the judge is uncalibrated), native self-service (the judge has not run), competitive standing (no neutral eligibility), temporal stability beyond one afternoon, the consumer apps (no telemetry volume and no human-assisted study), or other categories. The outstanding validation of Section 9, an external design partner in a different category, and a pre-registered confirmatory run on the locked test split are what stronger claims would need. A null result, a failed control and a disagreement with earlier summaries are all results here, and the instrument is more trustworthy for having produced them.

Statements

Data and code. The per-run data pack (without prompt text, which is private), the analysis code, the figure scripts, the generated number ledger with the provenance of every number, and the simulation code and output are available from the author on request (InvokeRank, 2026b). The published method is at <https://invokerank.com/methodology> (InvokeRank, 2026d). Competitors are named only as replayed public listings; no competitor server was called.

Provider terms. All runs used the providers’ sanctioned APIs. InvokeRank never automates chatgpt.com or claude.ai, and no consumer-app conversation was observed.

Table 14: Claims and their status on 6 October 2026. Supported: under the tested conditions. Provisional: supported by exploratory or unreplicated evidence. Not yet supported: the analysis has not been run. Rejected: contradicted by the data. Requires external replication: supported on the owned integration only.

Claim	Status	Evidence	What would change it
The nested funnel’s product is an exact probability	Supported	Derivation (Section A)	None (algebraic)
The published scoring path is reproducible from per-run data	Supported	Independent port (Section 7.1)	A disagreement on new data
The host’s own answer, not a competitor, is TapTax’s main alternative	Supported for selection; requires external replication	Section 6.3	Neutral eligibility; other categories
No-tool runs are self-service	Not yet supported	No native verdicts	The native judge
Executed does not predict Completed	Provisional	Section 6.4; uncalibrated judge	Gold set, κ , PPI
The hosts differ in completion	Not yet supported	Judge confounded with host	Both judges on both hosts
The runs respond to tool metadata	Supported	Blinded control	A failed blinded round
Hosts choose between identical servers by label	Supported (retired design)	Clone split, both orders	<code>clone_swap</code> round
Tool-description rewrites move selection	Rejected at this resolution	Table 9	Larger or confirmatory experiments
A server description moves ChatGPT API selection	Provisional	Replicated exploratory arms	Pre-registered test-split run
Identical runs vary beyond sampling error	Supported on one ChatGPT API pool only	Section 7.3	More pools over days
The published intervals have nominal coverage	Rejected: modest undercoverage at the TapTax calibration, severe for the roll-up when intents are alike	Simulation (Section 7.5)	Joint roll-up; random-effects model
Tool need predicts selection	Provisional	Permutation test over eight intents	Independent coding; native judge
Synthetic results predict consumer-app behaviour	Not yet supported	No T volume, no A study	Telemetry, assisted study
TapTax is better than its competitors	Not supported (not tested)	v1 eligibility	Neutral eligibility

Telemetry and privacy. TapTax’s telemetry view holds tool, client host, outcome, duration and hour, with no user identifiers, and excludes test accounts, including the sandbox account used for execution runs.

Use of AI. The analysis code, simulation, figures and manuscript were drafted by an AI agent (Claude, Anthropic) from the data pack and the internal documentation listed in the references, at the author’s direction; every number is generated from the data by code, not written by hand. The author is accountable for the content.

References

- Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, and Ameet Deshpande. GEO: Generative engine optimization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5–16. ACM, 2024. doi: 10.1145/3637528.3671900.
- Alan Agresti and Brent A. Coull. Approximate is better than “exact” for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126, 1998. doi: 10.1080/00031305.1998.10480550.

- Amine Allouah, Omar Besbes, Josué D. Figueroa, Yash Kanoria, and Akshit Kumar. What is your AI agent buying? evaluation, biases, model dependence, & emerging implications of agentic e-commerce. In *Proceedings of the ACM Web Conference 2026 (WWW)*, pages 8697–8700. ACM, 2026. doi: 10.1145/3774904.3792943.
- Solomon Amos. InvokeRank product requirements document, version 2.18, 2026. Internal document, sections 0.1, 2.4, 11 to 20, 36 and 38.1; available from the author on request.
- Per Kragh Andersen and Niels Keiding. Multi-state models for event history analysis. *Statistical Methods in Medical Research*, 11(2):91–115, 2002. doi: 10.1191/0962280202sm276ra.
- B. W. Ang. Decomposition analysis for policymaking in energy: Which is the preferred method? *Energy Policy*, 32(9):1131–1139, 2004. doi: 10.1016/S0301-4215(03)00076-4.
- B. W. Ang. LMDI decomposition approach: A guide for implementation. *Energy Policy*, 86: 233–238, 2015. doi: 10.1016/j.enpol.2015.07.007.
- Anastasios N. Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I. Jordan, and Tijana Zrnica. Prediction-powered inference. *Science*, 382(6671):669–674, 2023. doi: 10.1126/science.adf6000.
- Anastasios N. Angelopoulos, John C. Duchi, and Tijana Zrnica. PPI++: Efficient prediction-powered inference. *The Annals of Applied Statistics*, 20(3):2235–2252, 2026. doi: 10.1214/26-AOAS2215.
- Anthropic. MCP connector. <https://platform.claude.com/docs/en/agents-and-tools/mcp-connector>, 2026a. Claude Developer Platform documentation, Accessed 6 October 2026.
- Anthropic. Tool search tool. <https://platform.claude.com/docs/en/agents-and-tools/tool-use/tool-search-tool>, 2026b. Claude Developer Platform documentation (tool versions `tool_search_tool_regex_20251119` and `tool_search_tool_bm25_20251119`), Accessed 6 October 2026.
- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=hSyW5go0v8>.
- Berk Atıl, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, et al. Non-determinism of “deterministic” LLM system settings in hosted environments. In *Proceedings of the 5th Workshop on Evaluation and Comparison of NLP Systems (Eval4NLP)*, pages 135–148. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.eval4nlp-1.12.
- R. H. Baayen, D. J. Davidson, and D. M. Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412, 2008. doi: 10.1016/j.jml.2007.12.005.
- Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01.
- Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x.

- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, et al. Lessons from the trenches on reproducible evaluation of language models, 2024.
- Z. W. Birnbaum. On the importance of different components in a multicomponent system. Technical report, Laboratory of Statistical Research, University of Washington, Seattle; distributed by the Defense Technical Information Center, 1968.
- Bjarni Haukur Bjarnason, André Silva, and Martin Monperrus. On randomness in agentic evals, 2026. Presented at the ICLR 2026 Workshop AIWILD.
- Thierry Blankenstein, Jialin Yu, Zixuan Li, Vassilis Plachouras, Sunando Sengupta, Philip Torr, Yarin Gal, Alasdair Paren, and Adel Bibi. BiasBusters: Uncovering and mitigating tool selection bias in large language models. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL <https://openreview.net/forum?id=DEg4vvElYu>.
- Samuel R. Bowman and George Dahl. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.385.
- Pierre Boyeau, Anastasios Nikolas Angelopoulos, Tianle Li, Nir Yosef, Jitendra Malik, and Michael I. Jordan. AutoEval done right: Using synthetic data for model evaluation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 5276–5290. PMLR, 2025. URL <https://proceedings.mlr.press/v267/boyeau25a.html>.
- Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324, 1952. doi: 10.2307/2334029.
- N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25, 1993. doi: 10.1080/01621459.1993.10594284.
- Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2), 2001. doi: 10.1214/ss/1009213286.
- A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller. Robust inference with multiway clustering. *Journal of Business & Economic Statistics*, 29(2):238–249, 2011. doi: 10.1198/jbes.2010.07136.
- George Casella. An introduction to empirical Bayes data analysis. *The American Statistician*, 39(2):83–87, 1985. doi: 10.1080/00031305.1985.10479400.
- Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, et al. Chatbot Arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 8359–8388. PMLR, 2024. URL <https://proceedings.mlr.press/v235/chiang24b.html>.
- C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404.
- Robert J. Connor. Sample size for testing differences in proportions for the paired-sample design. *Biometrics*, 43(1):207, 1987. doi: 10.2307/2531961.

- Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM)*, pages 87–94. ACM, 2008. doi: 10.1145/1341531.1341545.
- A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, Cambridge, 1997. doi: 10.1017/CBO9780511802843.
- Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, pages 82895–82920, 2024. doi: 10.52202/079017-2636.
- Alex Deng, Ya Xu, Ron Kohavi, and Toby Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 123–132. ACM, 2013. doi: 10.1145/2433396.2433413.
- Allan Donner. A review of inference procedures for the intraclass correlation coefficient in the one-way random effects model. *International Statistical Review*, 54(1):67, 1986. doi: 10.2307/1403259.
- Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 11812–11829. PMLR, 2024. URL <https://proceedings.mlr.press/v235/du24h.html>.
- Bradley Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, Cambridge, 2010. doi: 10.1017/CBO9780511761362.
- Bradley Efron and Carl Morris. Data analysis using Stein’s estimator and its generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975. doi: 10.1080/01621459.1975.10479864.
- Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco, New York, 1978. ISBN 0-668-04721-6.
- Kazem Faghih, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. Tool preferences in agentic LLMs are unreliable. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 20965–20980. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1060.
- Xiang Fei, Xiawu Zheng, and Hao Feng. MCP-Zero: Active tool discovery for autonomous LLM agents, 2025.
- C. A. Field and A. H. Welsh. Bootstrapping clustered data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(3):369–390, 2007. doi: 10.1111/j.1467-9868.2007.00593.x.
- Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical Methods for Rates and Proportions*. Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken, NJ, third edition, 2003. doi: 10.1002/0471445428.
- Tiantian Gan and Qiyao Sun. RAG-MCP: Mitigating prompt bloat in LLM tool selection via retrieval-augmented generation, 2025.

- Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, Boca Raton, FL, third edition, 2013. doi: 10.1201/b16018.
- Shahriar Golchin and Mihai Surdeanu. Time travel in LLMs: Tracing data contamination in large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=2Rwq6c3tvr>.
- Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 79–90. ACM, 2023. doi: 10.1145/3605764.3623985.
- Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Mao-song Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.664.
- Zikang Guo, Benfeng Xu, Chiwei Zhu, Wentao Hong, Xiaorui Wang, and Zhendong Mao. MCP-AgentBench: Evaluating real-world language agent performance with MCP-mediated tools. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30888–30896, 2026. doi: 10.1609/aaai.v40i37.40347. Issue 37.
- Mohammed Mehedi Hasan, Hao Li, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E. Hassan. Model Context Protocol (MCP) tool descriptions are smelly! towards improving AI agent efficiency with augmented MCP tool descriptions, 2026.
- Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. Thinking Machines Lab: Connectionism (blog), 10 September 2025, 2025. URL <https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/>. Accessed 6 October 2026.
- Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model Context Protocol (MCP): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*, 35(10):1–37, 2026. doi: 10.1145/3796519.
- Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, non-parametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021. doi: 10.1214/20-AOS1991.
- Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. MetaTool benchmark for large language models: Deciding whether to use tools and which to use. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=R0c2qtalgG>.
- Joel Huber, John W. Payne, and Christopher Puto. Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of Consumer Research*, 9(1):90, 1982. doi: 10.1086/208899.
- InvokeRank. InvokeRank scoring and runner packages (packages/scoring, packages/runner), 2026a. Source code at release 38c5838, 6 October 2026; available from the author on request.
- InvokeRank. Data pack for “InvokeRank and the TapTax case study”: runs, prompt metadata, judgements, controls, costs and telemetry, 3 to 5 October 2026, 2026b. Read-only export of 6 October 2026, without prompt text; available from the author on request.

- InvokeRank. Measurement engine: operating guide (docs/measurement.md), 2026c. Internal document at release 38c5838, 6 October 2026; available from the author on request.
- InvokeRank. The InvokeRank methodology, method v1.0. <https://invokerank.com/methodology>, 2026d. Method frozen 5 October 2026. Accessed 6 October 2026.
- Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 375–385. ACM, 2021. doi: 10.1145/3442188.3445901.
- Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007): 453–461, 1946. doi: 10.1098/rspa.1946.0056.
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.389.
- Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7969–7992. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.495.
- Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. Always valid inference: Continuous monitoring of A/B tests. *Operations Research*, 70(3):1806–1821, 2022. doi: 10.1287/opre.2021.2135.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know, 2022.
- Robert E. Kass and Duane Steffey. Approximate Bayesian inference in conditionally independent hierarchical models (parametric empirical Bayes models). *Journal of the American Statistical Association*, 84(407):717–726, 1989. doi: 10.1080/01621459.1989.10478825.
- Leslie Kish. *Survey Sampling*. John Wiley & Sons, New York, 1965.
- Joel C. Kleinman. Proportions with extraneous variance: Single and independent samples. *Journal of the American Statistical Association*, 68(341):46–54, 1973. doi: 10.1080/01621459.1973.10481332.
- Ron Kohavi, Diane Tang, and Ya Xu. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, Cambridge, 2020. doi: 10.1017/9781108653985.
- Edward L. Korn and Barry I. Graubard. Confidence intervals for proportions with small expected number of positive counts estimated from survey data. *Survey Methodology*, 24(2):193–201, 1998. URL <https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X19980024356>.
- Aounon Kumar and Himabindu Lakkaraju. Manipulating large language models to increase product visibility, 2024.

- Robert J. Lavidge and Gary A. Steiner. A model for predictive measurements of advertising effectiveness. *Journal of Marketing*, 25(6):59–62, 1961. doi: 10.1177/002224296102500611.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.187.
- Moxin Li, Yong Zhao, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, Tat-Seng Chua, and Yang Deng. Knowledge boundary of large language models: A survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5131–5157. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.256.
- 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. URL <https://openreview.net/forum?id=i04LZibEqW>.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=zAdUB0aCTQ>.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65.
- Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher Pal, and Siva Reddy. AgentReward-Bench: Evaluating automatic evaluations of web agent trajectories. In *Second Conference on Language Modeling (COLM)*, 2025. URL <https://openreview.net/forum?id=fQcUZMPivu>.
- R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. John Wiley & Sons, New York, 1959.
- Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. MCP-Universe: Benchmarking large language models with real-world Model Context Protocol servers, 2025. Presented at the NeurIPS 2025 Workshop on Scaling Environments for Agents (SEA).
- Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks, 2024.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.546.
- David Manheim and Scott Garrabrant. Categorizing variants of Goodhart’s law, 2018.

- Daniel McFadden. Conditional logit analysis of qualitative choice behavior. In Paul Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, New York, 1974.
- Daniel McFadden. Modeling the choice of residential location. *Transportation Research Record*, 673:72–77, 1978. URL <https://trid.trb.org/view/87722>.
- Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.
- Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations, 2024.
- Guozhao Mo, Wenliang Zhong, Jiawei Chen, Qianhao Yuan, Xuanang Chen, Yaojie Lu, Hongyu Lin, Ben He, Xianpei Han, and Le Sun. LiveMCPBench: Can agents navigate an ocean of MCP tools? In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 9581–9592. ACM, 2026. doi: 10.1145/3770855.3817478.
- Model Context Protocol. Model Context Protocol specification, version 2026-07-28. <https://modelcontextprotocol.io/specification/2026-07-28>, 2026. Protocol specification, Accessed 6 October 2026.
- Suhong Moon, Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Woosang Lim, Kurt Keutzer, and Amir Gholami. Efficient and scalable estimation of tool representations in vector space, 2024.
- Carl N. Morris. Parametric empirical Bayes inference: Theory and applications. *Journal of the American Statistical Association*, 78(381):47–55, 1983. doi: 10.1080/01621459.1983.10477920.
- Fredrik Nestaas, Edoardo Debenedetti, and Florian Tramèr. Adversarial search engine optimization for large language models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL <https://openreview.net/forum?id=hkdqxN3c7t>.
- OpenAI. Tool search. <https://developers.openai.com/api/docs/guides/tools-tool-search>, 2026. OpenAI API documentation, Accessed 6 October 2026.
- Simon Ott, Adriano Barbosa-Silva, Kathrin Blagec, Jan Brauner, and Matthias Samwald. Mapping global dynamics of benchmark creation and saturation in artificial intelligence. *Nature Communications*, 13:6793, 2022. doi: 10.1038/s41467-022-34591-0.
- Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. An empirical study of the non-determinism of ChatGPT in code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–28, 2025. doi: 10.1145/3697010.
- Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pages 68772–68802, 2024. doi: 10.52202/079017-2197.
- Shishir Patil, Tianjun Zhang, Xin Wang, and Joseph Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, pages 126544–126565, 2024. doi: 10.52202/079017-4020.
- Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR, 2025. URL <https://proceedings.mlr.press/v267/patil25a.html>.

- Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.130.
- Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiushi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. SMART: Self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4604–4621. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.239.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=dHng200Jjr>.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Towards completeness-oriented tool retrieval for large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1930–1940. ACM, 2024. doi: 10.1145/3627673.3679847.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025a. URL <https://openreview.net/forum?id=QKBu1B0Awd>.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8), 2025b. doi: 10.1007/s11704-024-40678-2.
- Inioluwa Deborah Raji, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada. AI and the everything in the whole wide world benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/084b6fbb10729ed4da8c3d3f5a3ae7c9-Abstract-round2.html>.
- Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4), 2023. doi: 10.1214/23-STS894.
- J. N. K. Rao and A. J. Scott. A simple method for the analysis of clustered binary data. *Biometrics*, 48(2):577, 1992. doi: 10.2307/2532311.
- Marvin Rausand, Anne Barros, and Arnljot Høyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, Hoboken, NJ, 2020. doi: 10.1002/9781119373940.
- Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. BetterBench: Assessing AI benchmarks, uncovering issues, and establishing best practices. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*, pages 21763–21813, 2024. doi: 10.52202/079017-0685.
- Martin S. Ridout, Clarice G. B. Demétrio, and David Firth. Estimating intraclass correlation for binary data. *Biometrics*, 55(1):137–148, 1999. doi: 10.1111/j.0006-341X.1999.00137.x.
- Herbert Robbins. An empirical Bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 157–163, Berkeley, CA, 1956. University of California Press. doi: 10.1525/9780520313880-015.

- Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3391–3409. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.174.
- Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.722.
- Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models’ sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=RIu5lyNXjT>.
- L. S. Shapley. A value for n -person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, number 28 in Annals of Mathematics Studies, pages 307–317. Princeton University Press, Princeton, NJ, 1953. doi: 10.1515/9781400881970-018.
- Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren’t tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24497–24524. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1258.
- Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228. ACM, 2018. doi: 10.1145/3219819.3220088.
- J. G. Skellam. A probability distribution derived from the binomial distribution by regarding the probability of success as variable between the sets of trials. *Journal of the Royal Statistical Society: Series B (Methodological)*, 10(2):257–261, 1948. doi: 10.1111/j.2517-6161.1948.tb00014.x.
- Jonathan Sneh, Ruomei Yan, Jialin Yu, Philip Torr, Yarin Gal, Sunando Sengupta, Eric Sommerlade, Alasdair Paren, and Adel Bibi. ToolTweak: An attack on tool selection in LLM-based agents, 2025.
- Wei Song, Haonan Zhong, Ziqi Ding, Jingling Xue, and Yuekang Li. Help or hurdle? rethinking Model Context Protocol-augmented large language models, 2025.
- Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, Cambridge, second edition, 2009. doi: 10.1017/CBO9780511805271.
- Phat T. Tran-Truong and Xuan-Bach Le. Measuring the unmeasurable: Markov chain reliability for LLM agents, 2026.
- Vladimir Vovk and Ruodu Wang. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021. doi: 10.1214/20-AOS2020.
- Hanna Wallach, Meera Desai, A. Feder Cooper, Angelina Wang, Chad Atalla, Solon Barocas, Su Lin Blodgett, Alexandra Chouldechova, Emily Corvi, P. Alex Dow, et al. Position: Evaluating generative AI systems is a social science measurement challenge. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of

- Proceedings of Machine Learning Research*, pages 82232–82251. PMLR, 2025. URL <https://proceedings.mlr.press/v267/wallach25a.html>.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.511.
- Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. ToolGen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL <https://openreview.net/forum?id=XLMAmowdY>.
- Ruodu Wang and Aaditya Ramdas. False discovery rate control with e-values. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(3):822–852, 2022. doi: 10.1111/rssb.12489.
- Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10303–10315. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.691.
- Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Shah, Yujia Bao, et al. MCP-Bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026a. URL <https://openreview.net/forum?id=fe8mzHwMxN>.
- Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. MCPTox: A benchmark for tool poisoning on real-world MCP servers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35811–35819, 2026b. doi: 10.1609/aaai.v40i42.40895. Issue 42.
- Zihan Wang, Rui Zhang, Yu Liu, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, Hongwei Li, and Guowen Xu. MPMA: Preference manipulation attack against Model Context Protocol. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 35838–35846, 2026c. doi: 10.1609/aaai.v40i42.40898. Issue 42.
- Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556, 2025. doi: 10.1162/tacl_a_00754.
- D. A. Williams. The analysis of binary responses from toxicological experiments involving reproduction and teratogenicity. *Biometrics*, 31(4):949, 1975. doi: 10.2307/2529820.
- Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, et al. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL <https://openreview.net/forum?id=uobR0wBsJm>.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

- Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL <https://openreview.net/forum?id=3GTtZFiajM>.
- Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don’t know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.551.
- Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, pages 188045–188077, 2025. doi: 10.52202/085713-5653.
- Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, et al. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11388–11422. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.637.
- Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024a. URL <https://openreview.net/forum?id=shr9PXz7T0>.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), Datasets and Benchmarks Track*, pages 46595–46623, 2023. doi: 10.52202/075280-2020.
- Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. ToolRerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16263–16273. ELRA and ICCL, 2024b. URL <https://aclanthology.org/2024.lrec-main.1413>.

A The posterior of Q

The chain rule for nested events. Let $D \supseteq S \supseteq I \supseteq E \supseteq C$ be events of one run. For nested events $P(S \mid D) = P(S \cap D)/P(D) = P(S)/P(D)$, and in general $P(A_k \mid A_{k-1}) = P(A_k)/P(A_{k-1})$. The product telescopes:

$$r_D r_S r_I r_E r_C = P(D) \cdot \frac{P(S)}{P(D)} \cdot \frac{P(I)}{P(S)} \cdot \frac{P(E)}{P(I)} \cdot \frac{P(C)}{P(E)} = P(C). \quad (5)$$

No independence between stages is assumed. The identity is algebraic and holds whenever every $P(A_{k-1}) > 0$.

Mixing over prompts. Let a run’s prompt x be drawn from a population with distribution F , and let $\pi_k(x) = P(A_k \mid A_{k-1}, x)$. The marginal stage rates are $r_k = P(A_k \mid A_{k-1}) = E_F[\prod_{j \leq k} \pi_j(x)]/E_F[\prod_{j < k} \pi_j(x)]$, and their product telescopes to $E_F[\prod_j \pi_j(x)] = P(C)$, the marginal probability of completion. So Q is a population quantity: the probability that a request drawn from the corpus’s prompt population, run once, completes through the integration. Pooled

counts estimate each marginal r_k consistently when every prompt contributes runs in proportion to its weight in F . With equal K per prompt, prompts are weighted equally. The product of the marginal rates is therefore the mean of the per-prompt products $\prod_k \pi_k(x)$; it differs from the product of the mean per-prompt rates, $\prod_k E_F[\pi_k(x)]$, unless the $\pi_k(x)$ are uncorrelated across prompts.

Factorisation of the likelihood. For one cell with independent runs, the sequential counts give the likelihood

$$L(r_D, \dots, r_C) = \prod_{k \in \{D, S, I, E, C\}} r_k^{y_k} (1 - r_k)^{n_k - y_k}, \quad n_k = y_{k-1}, \quad n_D = n, \quad (6)$$

which factorises in the r_k . With independent priors, the posteriors are independent and conjugate: $r_k \mid \text{data} \sim \text{Beta}(a_k + y_k, b_k + n_k - y_k)$. Hence $E[Q \mid \text{data}] = \prod_k E[r_k \mid \text{data}]$ exactly, which is how the published path computes the point estimate, and the interval follows from multiplying independent draws. Posterior independence is therefore a property of this model, not an added assumption.

Where it can fail. The factorisation needs the runs to be independent given the r_k . With prompts as clusters it does not hold: two runs of one prompt share $\pi_k(x)$. The published path responds by deflating counts by a design effect, which corrects the variance of each stage’s pooled proportion under an exchangeable correlation model but treats the stages separately. If prompts that surface easily also select readily ($\text{Cov}_F(\pi_D, \pi_S) > 0$), the estimators $\hat{r}_D$ and $\hat{r}_S$ covary across samples of prompts, and the variance of $\hat{r}_D \hat{r}_S$ differs from the product of independent stage variances. Section 7.5 measures the crude prompt-level correlation of surfacing and selecting at 0.09 and 0.10; the model-based latent correlation is larger (0.89 and 0.30), and the simulation of Section H finds that coverage barely changes between scenarios with latent correlation 0, 0.5 and the fitted value.

The design effect. For a proportion estimated from m_p runs of each of P prompts with exchangeable intra-prompt correlation ρ , $\text{Var}(\hat{r}) = \frac{r(1-r)}{\sum_p m_p} [1 + (\bar{K} - 1)\rho]$ with $\bar{K} = \sum_p m_p^2 / \sum_p m_p$ (Kish, 1965; Rao and Scott, 1992). Deflating (y, n) by this factor gives a beta posterior whose variance matches the cluster-corrected variance to first order; this is an approximation, not a likelihood, and it degrades when ρ is estimated poorly (Section 7.4).

Roll-up. The roll-up $\sum_c w_c Q_c$ is the probability that a request drawn from the intent mixture w completes, so it is an arithmetic mean of probabilities. Combining the intents’ posterior draws index by index treats the intents as independent a posteriori. With a shared prior fitted from the same data they are dependent, weakly when the intents differ and strongly when they look alike, because the capped prior then makes every intent’s posterior nearly the pooled-data posterior; the simulation in Section H shows the resulting roll-up intervals covering 65.0% of the time at a nominal 90%.

B Loss share

Definition and exactness. From Equation (2), $-\ln Q = \sum_k -\ln r_k$, a sum of non-negative terms. The loss share $L_k = -\ln r_k / \sum_j -\ln r_j$ is each stage’s fraction of the total log loss; the shares sum to one by construction. It is the unique decomposition of $-\ln Q$ that is additive over stages and attributes to each stage only its own rate. Because the log-loss is additive, the Shapley attribution of $-\ln Q$ over stages, treating each stage’s rate as a player that moves from 1 to r_k , assigns exactly $-\ln r_k$ to stage k (Shapley, 1953); the loss share is the normalised

Shapley value of the log loss. It is the logarithmic index decomposition of a product of factors with a perfect funnel as the base (Ang, 2004, 2015), renamed; its application to agent tool-use funnels has no precedent that we found [UNVERIFIED].

Edge cases. As $r_k \rightarrow 1$, $-\ln r_k \rightarrow 0$ and the stage’s share vanishes. If every stage is near one, $r_k = 1 - \epsilon_k$ with small ϵ_k , then $-\ln r_k \approx \epsilon_k$ and $L_k \approx \epsilon_k / \sum_j \epsilon_j$: the shares become ratios of small failure rates and inherit their relative sampling error, which is large. If any $r_k = 0$, the log loss is infinite; the implementation then gives the whole share to the zero stages, split equally. Loss share is diagnostic: it says where the probability is lost under the current rates, not what would happen if a stage were changed.

TapTax. For the baseline’s pooled rates, Surfaced carried 44.0% of the log loss on the Claude API and Selected 56.0%; on the ChatGPT API 42.1% and 57.9%; Invoked carried none on either host. Because Surfaced means a tool on one host and a server on the other, the split between D and S is not comparable across hosts.

C Lever

Definition. For stage k with posterior mean $\tilde{r}_k$ and reference rate r_k^{ref} (the median over the integration’s intents, or the best competitor’s rate), write $Q_{-k} = \prod_{j \neq k} \tilde{r}_j$. Then

$$\text{Lever}_k = 100 [Q_{-k} \min(1, r_k^{\text{ref}}) - \tilde{Q}]^+ = 100 \tilde{Q} \left(\frac{\min(1, r_k^{\text{ref}})}{\tilde{r}_k} - 1 \right)^+, \quad (7)$$

the change in InvokeRank points if stage k alone moved to the reference with every other stage fixed. When $\tilde{r}_k = 0$ the implementation returns $100 r_k^{\text{ref}} Q_{-k}$, the limit of the first form.

Properties. Levers are not additive: moving two stages to their references multiplies their factors, so the joint gain exceeds the sum of the two levers whenever both are positive. A lever is zero for any stage already at or above its reference, however low it is in absolute terms. It depends on the reference: the intent median measures a stage against the integration’s own other intents, which conflates intent difficulty with integration quality; the best competitor’s rate requires valid competitor rates, which need neutral eligibility. It is an accounting identity over posterior means, a “what if” uplift, and not a treatment effect: nothing guarantees that any available intervention can move stage k alone, or move it at all. Algebraically $\text{Lever}_k = 100 Q_{-k} (r_k^{\text{ref}} - \tilde{r}_k)^+$: the Birnbaum importance of component k in a series system times the gap to the reference (Birnbaum, 1968), which with $r^{\text{ref}} = 1$ is the reliability-engineering *improvement potential* (Rausand et al., 2020), and in general a one-factor-at-a-time counterfactual of index decomposition (Ang, 2004). It is a re-application, not a new quantity.

Native self-service. Runs judged as correct native self-service are excluded from the Lever’s rates unless the customer declares tool use necessary for correctness, freshness, private state or an action. The reason is that a low selection rate where the host already meets the need is a distribution reality, not a metadata defect. Because no native verdicts exist, the exclusion has no effect on the TapTax figures in Section 6.7.

The first test. The 3 October candidate is the first intervention whose realised effect can be set against the Lever (Section 6.7). Its targets carried levers of 28.3 and 12.0 points at Selected on the Claude API and of 49.1 and 48.6 at Surfaced on the ChatGPT API; no realised per-intent change was distinguishable from zero. A fair test of the Lever needs interventions chosen in advance to target the stage it names, with enough prompts per intent to resolve changes of the predicted size.

D Invoke Share and Selection Share

Definitions. For a contested intent c with independently eligible set $\mathcal{E}_c$, $|\mathcal{E}_c| \geq 2$,

$$IS_{ic} = \frac{Q_{ic}}{\sum_{j \in \mathcal{E}_c} Q_{jc}}, \quad SS_{ic} = \frac{r_{D,ic} r_{S,ic}}{\sum_{j \in \mathcal{E}_c} r_{D,jc} r_{S,jc}}, \quad (8)$$

and the roll-up is $\sum_c w_c IS_{ic}$ over contested intents, with $IS_{ic} = 0$ where i is not eligible, so that shares sum to one across integrations. If any eligible integration of any contested intent lacks Executed or Completed, every intent is computed as Selection Share, so a roll-up never mixes the two. The no-tool path and host built-in tools are never in the denominator; the Agent Resolution Split reports them.

What the share is. With replayed competitors the selection-level quantity $r_{D,ic} r_{S,ic}$ is the probability that integration i is the first external call for a request of intent c . Summed over the eligible set it is the probability that *some* eligible integration is chosen first, so SS_{ic} is the conditional probability that i is chosen given that an eligible integration is chosen. It is not market share, and it is not share of all resolutions; it is a share of first choices among a defined set, a re-application of share-of-choice measures from discrete choice [UNVERIFIED].

Compositional dependence. Within one run at most one integration is chosen first, so the first-choice indicators of different integrations are mutually exclusive and their probabilities negatively dependent. The published path computes each integration’s Q_{ic} posterior separately and forms shares from independent draws. Independent draws ignore the constraint $\sum_j P(\text{first choice} = j) \leq 1$ and the negative covariance it implies, so they overstate the variance of the denominator and misstate each share’s interval. A joint model fixes this: per intent, the first-choice outcome over $\mathcal{E}_c \cup \{\text{other, none}\}$ is categorical, with a Dirichlet posterior on the cell probabilities (deflated for prompt clustering), from which shares are a deterministic transform of each draw. That model also yields the Agent Resolution Split and Invoke Rating’s outside option from the same posterior.

Status. Neither share has been computed for TapTax: no neutral eligibility exists, and the v1 competitor rates stay internal.

E Prompt taxonomy and corpus construction

Generation. For each intent, both model families (Claude Sonnet 5.5 and GPT-6.1 Sol) wrote prompts from the intent’s definition with structured output, across prompt classes, and a canary string marks the corpus. Prompts within cosine 0.9 of another under a sentence embedding were dropped, and a brand-name list excluded product names. The cost ledger in the data pack records \$0.51 for corpus generation and embedding calls.

Classes. The MTD corpus holds 153 indirect prompts (the need is described, not the tool), 37 persona prompts, 19 boundary prompts, 15 multi-step prompts, 11 write-action prompts, 5 competitive prompts and 80 negatives. The corpus has 0 branded prompts and 0 flagged near-duplicates. Positive prompts carry an expected behaviour (call an eligible tool, or ask a clarifying question); negatives expect no tool.

Splits. Splits are made by seed cluster (208 clusters), targeting 50% development, 30% validation and 20% test, so that paraphrases never straddle splits. The realised split is 182, 79 and 59 prompts. The development split informed the 3 October description candidate, so experiments are evaluated on the validation split; the test split has never been run.

Realism review. A transparent phrase rule flags prompts that depend on context the harness does not supply: an attachment, an email, a document, a screenshot, “the above” or “this record”. A prompt whose harness context is non-empty is never flagged. The review sample is every flagged prompt plus a stratified 10% of each intent and class; prompts reviewed unrealistic leave the next corpus revision, and every figure can still be reported on the original corpus. On 6 October 2026 the sample of 91 prompts is drawn and 0 are reviewed.

Blind relabelling of negatives. A negative becomes *optional* only when both model families, shown the prompt text alone (never a run, a tool list or a product), judge that using an app or calculator would be reasonable. On the calculator corpus this moved 32 negatives to optional; on the MTD corpus it has not run. Where the label matters, Table 9 and Figure 5 report false invokes on the original labels.

The calculator corpus. The second corpus, `uk-tax-calculations`, has 10 calculator intents in three groups (pay and companies, property and investments, VAT and penalties) and 400 prompts, 300 positive, 68 negative and 32 optional after the relabel. It is also an instrument-validation category, not a benchmark, and it is measured against TapTax 1.1 with 48 tools.

F Eligibility: v1 and the neutral process

v1 (used in this paper). The suite `taptax-uk` takes its intents from a suite file written for TapTax, and every prompt of those intents is positive for every pool member. Competitors are scored on the same intents whatever their tools can do. This is adequate for validating the instrument on TapTax and invalid for comparing products.

The neutral process (built, not run).

1. **Capability union.** Every materially relevant indexed integration in the category, one per brand, with its pinned snapshot’s tool names, descriptions and schemas and its listings, each capability carrying its provenance.
2. **User language.** Search queries, support requests, community questions and interviews, each with its source; first-party demand shapes intents but is never eligibility evidence, and lines that name a product are withheld from the models.
3. **Intent proposals.** Both model families propose intent definitions and success criteria from the union and the user language. A deterministic neutrality lint rejects a definition that contains a product name, a tool identifier, or a word 4-gram found in exactly one integration’s descriptions and never in the user language. Proposals are merged across families; the owner accepts or excludes them.
4. **Eligibility.** For every integration and intent, both families judge from that integration’s own tools and listing. A quoted tool must exist in the pinned snapshot and its quote must appear in that tool’s description, or the claim is dropped; an eligible claim with no surviving quote becomes unknown. Families combine by agreement only: both eligible gives eligible, both ineligible gives ineligible, anything else unknown.
5. **Review.** A person approves, rejects or edits each row; every change is versioned in an append-only audit log, and a rerun never overwrites a reviewed row. Public shares use reviewed rows and contested intents only.

Status. A dry run of the union for “UK sole trader and landlord accounting” found 18 integrations and 377 capabilities, including non-UK tools that the owner must exclude by hand. No build, eligibility row or review exists.

Recommended sensitivity analysis. Eligibility is the denominator of every share, so the share’s stability under alternative defensible eligibility matrices is part of its uncertainty. We recommend reporting, for each contested intent, the rank order and shares under (a) reviewed rows only, (b) unknown treated as eligible, (c) unknown treated as ineligible, and (d) each model family’s judgement alone; and the fraction of integration-intent pairs whose status changes between (b) and (c). An independent annotator, not the owner of any measured integration, should review a stratified sample, with Cohen’s κ against the owner’s review.

G Instrument-control details

Every control arm replays the base suite’s exact pinned versions, corpus, surfaces and runner models, with the same prompts, repeats and rotations, queued interleaved prompt by prompt. Control copies are integrations with a **control** status: never public, never probed, never counted, and wearing the focal integration’s measurement label unless the control changes the label.

Placebo. Two arms identical to the base suite are run together; the check holds when the 95% paired prompt-bootstrap interval of their difference covers zero. At the 95% level a sound instrument fails about one round in twenty, so a failure is re-run on fresh prompts before it is believed. The reference arm is shared with the blinded check.

Blinded metadata. The focal integration is replaced by a copy with tool names replaced by neutral identifiers (`tool_01` and so on), and titles, descriptions, schema text, server instructions and server description removed; the label, argument names and types are kept. The check holds when selection falls by at least half and the 95% upper bound of the change is below zero.

Decoy. An unrelated public integration is added to the pool. The check counts prompts, not runs, and fails only when the 90% interval’s lower bound exceeds 2%. With zero picks on n prompts the Jeffreys upper bound is the 95th percentile of $\text{Beta}(1/2, n + 1/2)$: 3.2% for $n = 58$ and 8.6% for $n = 21$. The rule passes on any sample with no picks; the bound is what the sample can vouch for.

Clone split (retired). An exact copy of the focal integration is listed as `<label>-2` beside the original. The check holds when the copy’s order-balanced share of the pair’s first picks covers one half. Label and identity are confounded, so a failure cannot distinguish label bias from plumbing asymmetry.

Clone symmetry (clone_swap, built, not run). Copy A is the focal integration at its pinned version and copy B its exact control copy: byte-identical tools, instructions, server description and server information. Both wear matched opaque labels: equal length, the shape `srv-` plus four consonants or digits, never a word, distinct from every other label. Run k on prompt i lands in cell $(i + k) \bmod 4$ of a 2×2 : bit 0 lists copy B first, bit 1 swaps the labels. With $K = 2$ every prompt sees both listing orders. Every run records the labels the host saw, and tool references are resolved against them. The check reports copy B’s share of the pair averaged over the four cells, with a prompt-bootstrap 90% interval, and holds when it covers one half. It also reports the label bias (the share of whichever copy wears the first label, minus one half, order-balanced) and the position bias (the share of whichever copy is listed first, minus one

half, label-balanced), and their largest bound as the *resolution limit*: near-identical competitors closer than that in pair share cannot be told apart. A round on the TapTax validation split needs about 158 runs per surface.

Real usage. The synthetic tool mix of the base suite (first calls in selection runs; every call in execution runs) is compared with the integration’s real calls on the same host over 90 days, by Spearman rank correlation with a bootstrap interval; it holds when the lower bound is above zero. It needs at least 30 real calls. A low correlation has two readings, that the corpus misses real demand or that the API and the consumer app differ, which only intent-level telemetry or human-assisted runs can separate.

H Statistical sensitivity analyses and the coverage simulation

Sensitivity of the headline to ρ . Forcing every stage’s ρ to one value and rerunning the published path on the baseline moves the point estimate by less than half a point and changes the interval’s width. On the Claude API the 90% interval is 37.6 to 44.6 at $\rho = 0$ (width 6.9), 36.7 to 45.7 at the within-intent ρ_S (width 8.9) and 36.4 to 46.0 at $\rho = 1$ (width 9.5), against the published 37.1 to 46.1 (width 9.0). On the ChatGPT API the widths are 6.1, 7.6 and 7.7, against the published 7.5 and the prompt-cluster bootstrap’s 9.3. Even counting every prompt once, the model interval on the ChatGPT API is narrower than the bootstrap’s. The remaining difference comes from partial pooling: the shared prior for Selected has $\hat{\kappa} = 14.9$ on the ChatGPT API against 3.2 on the Claude API, so the ChatGPT intents borrow more strength from each other, which narrows the roll-up if the intents really are exchangeable.

Design of the simulation. The simulation asks whether the published intervals for the selection-level quantity attain their nominal coverage under data-generating processes calibrated to the TapTax baseline (InvokeRank, 2026b). Each simulated study is one suite execution: the baseline’s prompts per intent, K runs per prompt and one session effect shared by every run. Prompt propensities are bivariate logit-normal with intent fixed effects, prompt-level standard deviations for Surfaced and Selected given Surfaced, and a latent cross-stage correlation, all fitted by maximum likelihood to the baseline per surface; the fitted latent correlation is 0.89 on the Claude API (95% Wald interval -0.04 to 0.99) and 0.30 on the ChatGPT API (-0.41 to 0.79), much larger on the Claude API than the crude correlation of per-prompt shares in Section 7.5, which is attenuated by $K = 2$. A predictive check reproduces the baseline’s pooled ρ , discordant-prompt counts and stage rates within the 5th to 95th percentiles of the simulated baselines. The between-run SD is fitted per stage from the replicated identical arms with prompt fixed effects; its maximum-likelihood value is zero on both surfaces, and scenarios labelled “run SD upper” use its 95% profile upper bound (1.08 and 0.43 on the logit scale for Surfaced and Selected on the Claude API, 0.71 and 0.69 on the ChatGPT API). In all, 38 scenarios vary one factor at a time from the calibrated centre (cross-stage correlation, K , prompts per intent, number of intents, beta-distributed propensities, the run SD, two sessions per study) plus the scoring test’s own data-generating process (three identical intents), with 1,000 studies each, so the Monte Carlo standard error of a coverage near 90% is about one point. The estimators are the published path called through the scoring package itself (*capped*), the same path without the prior cap (*plain empirical Bayes*), the published path with pooled ρ , a prompt-cluster bootstrap of the plug-in estimate, a generalised linear mixed model of the same bivariate logit-normal family with intent fixed effects (posterior mode under weak priors; delta-method intervals), and, for two-session studies, a run-and-prompt bootstrap. Figure 10 shows coverage of nominal 90% intervals for the roll-up and for the per-intent intervals pooled over intents.

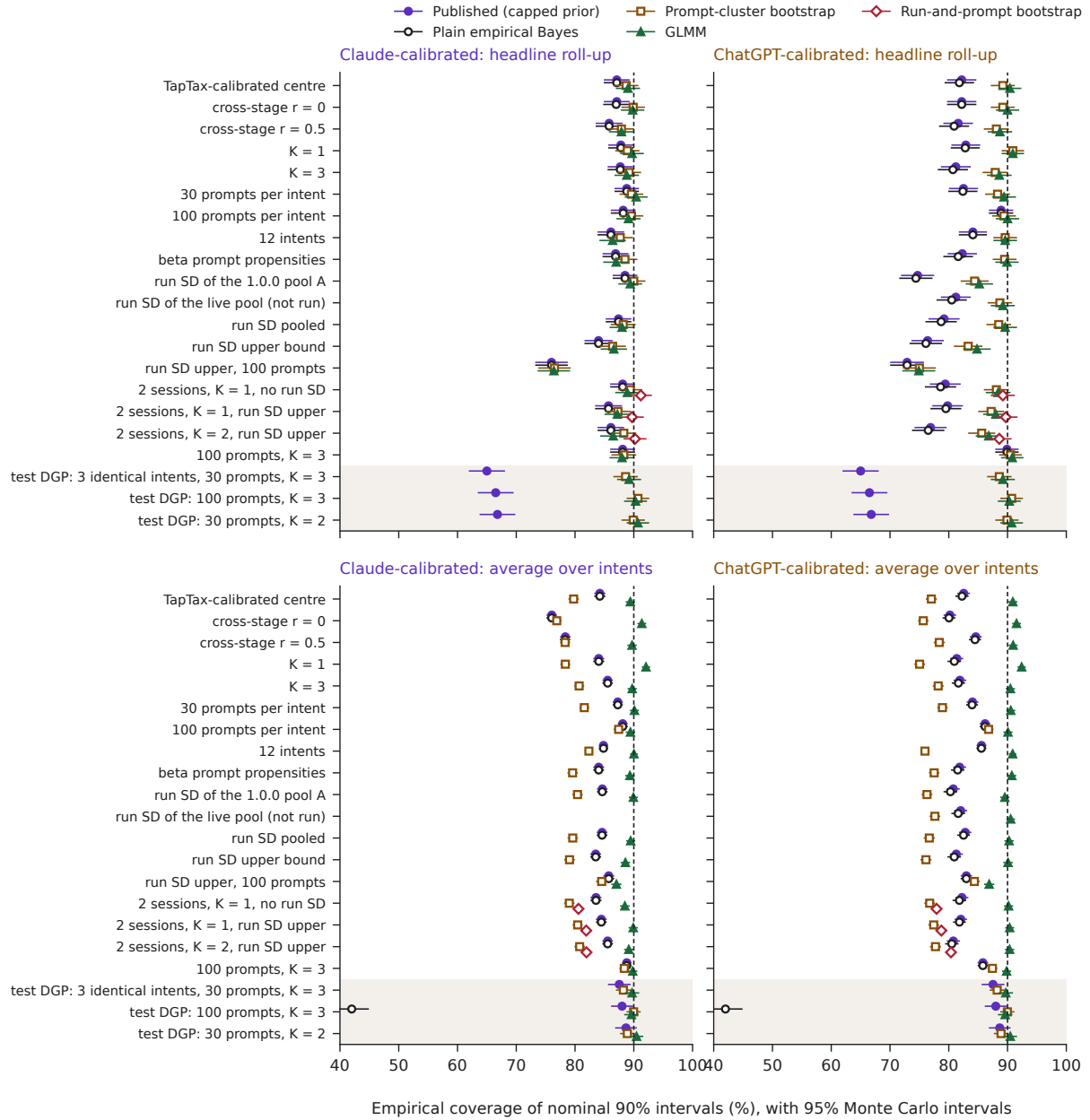


Figure 10: Empirical coverage of nominal 90% intervals for the selection-level quantity, by scenario and estimator, with 95% Monte Carlo intervals; 1,000 studies per scenario. Top: the equal-weight roll-up (the headline). Bottom: per-intent intervals pooled over intents. Shaded: the scoring test's data-generating process (three identical intents). Source: `data/sim/coverage.csv`, produced by `scripts/sim`; the published estimator is the scoring package itself, called through a TypeScript harness.

Results. In the Claude-calibrated centre, the published method’s 90% roll-up intervals covered 87.1% of the time (Monte Carlo SE 1.1) and its per-intent intervals 84.2%; in the ChatGPT-calibrated centre, 82.2% and 82.5%. The cap made no difference at the TapTax calibration: plain empirical Bayes covered 87.1% and 81.8%. The mixed model covered 89.0% and 90.4% of roll-ups and 89.4% and 90.9% of single intents, close to nominal. The prompt-cluster bootstrap covered roll-ups well (88.7% and 89.2%) and single intents poorly (79.8% and 77.0%), as percentile bootstraps do with two dozen clusters. The single-stage intervals the published path displays, Jeffreys on deflated counts, held their level: at Surfaced they covered 89.8% and 92.4%, and at Selected given Surfaced 91.0% and 90.5%. Without deflation the Surfaced intervals covered only 77.0% and 78.7%, so the design effect is doing necessary work; this answers the question of how Jeffreys intervals behave on fractionally deflated counts for these designs, for which we found no published study.

Under the scoring test’s own design, three intents with identical true rates, the published method’s per-intent intervals covered 87.5%, consistent with the test’s claim, but its roll-up intervals covered only 65.0%. When intents look alike the cap binds, each intent’s posterior becomes nearly the pooled-data posterior, and the roll-up then averages draws that share most of their information as if they were independent. Plain empirical Bayes, the method the cap replaced, covered 42.0% of per-intent intervals at 100 prompts per intent, reproducing the failure that motivated the cap. For the full five-stage InvokeRank under the same design, the published method covered 86.8% of per-intent intervals and 65.7% of roll-ups.

A run-level component degrades every single-session method. With the run SD at its upper bound, the published roll-up covered 84.0% and 76.4%, and at 100 prompts per intent 76.0% and 72.9%: more prompts narrow the interval without touching the run component. The mixed model, which has no run effect, fell to 76.4% and 74.9% in the same cells. At the run SD fitted to the ChatGPT API’s first replicate pool (Section 7.3), the published roll-up covered 74.7%, the prompt-cluster bootstrap 84.4% and the mixed model 85.2%; at the Claude API’s fitted value, which is zero, they covered 88.5%, 90.0% and 89.4%. With two sessions per study, the run-and-prompt bootstrap recovered nominal roll-up coverage (90.2% and 88.6%) where the published path did not (86.1% and 76.9%).

What follows. Three changes would bring the published intervals to nominal under every scenario we simulated: compute the roll-up from a joint model (or from the pooled-data posterior directly) rather than from independent per-intent draws; replace the per-cell ρ and the deflated beta posteriors with a prompt random-effects model; and, wherever the run SD may be non-zero, run at least two sessions and resample them. Until then the published roll-up interval should be read as somewhat too narrow, more so on the ChatGPT API and when intents are alike.

I Additional TapTax experiments and the full ledger

The body of the paper reports the pilot, baseline, repeat, execution runs, the 3 October metadata experiment and the control round. Between 4 and 5 October the programme ran a further set of exploratory arms, most of them chosen after seeing earlier results. We list them all so that the number of comparisons is visible (Table 5); Table 9 gives every paired result. All are on the validation split, with $K = 2$ and 5 rotations, and all are S evidence.

The calculator corpus. A second corpus of 10 calculator intents in three groups (pay and companies; property and investments; VAT and penalties) was measured against TapTax 1.1 with its own owner-approved competitor sets per group (Section E). At its first run on 4 October, TapTax was selected on 44.8% of pay, 26.0% of property and 22.7% of VAT positive runs on the Claude API, and 34.5%, 2.0% and 15.9% on the ChatGPT API, where competitors took the

first call in 10, 20 and 17 runs respectively. On the Claude API no competitor was ever the first call on this corpus.

Tool inventory. Measurement-only copies of 1.1.0 kept 15 or 23 of its 48 tools (**keep15**, **keep23**), and two split-server arms moved the calculators to a second measurement-only server (in two naming variants) whose picks count as TapTax’s. Against the two 1.1.0 runs on the same pool, none of these arms produced a change whose interval excluded zero on the Claude API.

Account wording and disambiguation. Rewritten descriptions on five account-reading tools gave +5.2 points on the Claude API (−1.7 to +12.1) and, pooling two replicates on the ChatGPT API, +3.9 (−0.9 to +9.9). The ChatGPT API replicate pair is the case where a prompt-only bootstrap on the pooled runs excluded zero while the run-and-prompt bootstrap did not; the method now uses the latter whenever an arm is replicated. A candidate with disambiguated descriptions (1.1.1) gave −1.7 (−6.9 to +3.4).

Server-description arms (OpenAI path only). Measurement-only copies of the live version differed only in `server_description`: one naming the calculators, one with a second wording that included figures, and one combining the first with dated tax-rule facts in the calculator tool descriptions. Their results are in Section 6.7: large lifts on the pay and property corpora, a smaller one on VAT, none on MTD, and higher false invokes on the property corpus. A further arm added the dated tax-rule facts to the tool descriptions alone; it changed nothing on the ChatGPT API, and on the Claude API it gave +4.9 points on the VAT corpus (0.0 to +12.1).

Misroute fix and mileage wording. Five descriptions reworded to stop tools being misrouted (Claude API only, because the ChatGPT API account had no credit at the time), and a mileage tool rewording (ChatGPT API only), produced no change distinguishable from zero.

Tool fit. Because Selected is integration-level, a 48-tool server can be “selected” with the wrong tool. The working sessions classified each pick as fitting or wrong using fit rules written partly with the TapTax side and partly decided by blind two-family prompt labels (whether a prompt concerns the Construction Industry Scheme, a company, pricing, or the person’s own saved records). Those labels are not in the data pack, so we report the tools chosen (Section 6.7) and not fit rates.

Version watch. TapTax is probed hourly; a new version triggers paired old-against-new suites on the validation split for each corpus, with an automatic replicate when the first pass excludes zero. Between 4 and 5 October it compared 1.0.0 with 1.1.0, 1.1.0 with the live 1.1 release and the live release with a later one that added one optional input. Only the first comparison’s first pass excluded zero on the Claude API (Section 6.7); the later releases changed nothing measurable (+3.4 and +1.7 points on the Claude API).

Replicates. To study run-to-run variance the programme scheduled eleven replicates of the live version on each API and twelve of 1.0.0 on the Claude API; nine, seven and six of them completed in full before the provider accounts ran out of credit (Section 7.3). Some suites did not complete, and they are excluded from every comparison:

- `taptax-x-uk-live-r11` and `taptax-x-uk-live-r12`;
- `taptax-x-uk-desc-c` to `taptax-x-uk-desc-l`;

- `taptax-x-uk-claude-live-r9` to `taptax-x-uk-claude-live-r12`;
- `taptax-x-uk-v100-g` to `taptax-x-uk-v100-l`.

The suites ending `-void` were abandoned comparisons.