UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

DÉPARTEMENT DES SCIENCES ADMINISTRATIVES

WORKING PAPER NO. 29

Protocol-Guided Autonomous Research with Claude Code

Scientific Verification, Reproducibility, and the Cost of Delegation: A Controlled Study

Simon-Pierre Boucher

Département des sciences administratives
Université du Québec en Outaouais

simon-pierre.boucher@uqo.ca

Gatineau – Pavillon Alexandre-Taché
283, boulevard Alexandre-Taché
Gatineau, Québec, Canada J9A 1L8

October 2026

Version 1.0

Abstract

This paper examines … We find that … Our results suggest … These findings contribute to the literature on …

Keywords: autonomous research agents; Claude Code; research protocols; executable verification; reproducibility; agent evaluation; delegation

JEL Classification: C18, C63, C88, O33

1. Related work

Eight bodies of work bear on whether the structure of the instructions given to an autonomous coding agent changes the scientific validity of its output, and at what cost. We read them as an argument: autonomous research systems have been evaluated, but never under a design that isolates the instruction regime (§1.1); benchmarks supply executable oracles that must be protected from the executor (§1.2); reproducibility research shows what re-execution buys (§1.3); preregistration gives the template for a protocol fixed before execution (§1.4); human–automation research rules out self-reported efficiency (§1.5); durable-execution research supplies the measurement apparatus (§1.6); and claim verification motivates the citation gates (§1.7). Section 1.8 states what this study adds.

1.1. Autonomous research systems and what their evaluations established

Tool-using autonomous experimentation was first demonstrated in chemistry: a GPT-4 agent with search, code execution and laboratory-automation tools performed six tasks including palladium-catalysed cross-coupling optimisation, with a physical oracle and human supervision (Boiko et al., 2023), and ChemCrow's 18 tools executed syntheses evaluated by both LLM and expert assessment (M. Bran et al., 2024).

End-to-end systems followed. The AI Scientist chains idea generation, code editing, experiments, plotting, paper writing and a simulated review; the preprint reports a cost below $15 per paper and an automated reviewer with near-human agreement on scores (Lu et al., 2024), and the line of work has since appeared in *Nature* (Lu et al., 2026). Its second version replaces human-written code templates with a progressive agentic tree search and a vision-language feedback loop on figures; its validity evidence is one of three autonomous manuscripts scoring above the average acceptance threshold at an ICLR workshop (Yamada et al., 2025). Agent Laboratory stages literature review, experimentation and report writing with optional human feedback at each stage; stage-wise feedback significantly improved research quality and expenses were 84% lower than previous autonomous methods (Schmidgall et al., 2025). data-to-paper walks agents from annotated data to a manuscript while programmatically back-tracing the flow of information so that a human can verify every number, in autopilot or copilot mode (Ifargan et al., 2025). Co-Scientist generates, critiques, ranks and evolves hypotheses in a tournament loop, with wet-laboratory validation performed downstream by collaborators (Gottweis et al., 2026); the Virtual Lab has an LLM principal investigator run meetings with specialist agents and a critic while humans guide the agenda and run the experiments (Swanson et al., 2025); Kosmos runs for up to twelve hours and about 200 rollouts over a shared world model, with accuracy judged by expert reading (Mitchener et al., 2025).

These evaluations established that end-to-end pipelines can produce reviewable papers cheaply,

that staging plus human feedback improves outcomes and lowers cost, and that traceable evidence chains can be built in. They did not establish whether the *structure* of the instruction—as distinct from the model, the scaffold or the amount of human co-piloting—changes validity: each system is a single pipeline without a control arm, any reviewer is in-system, and the outcome is a reviewer score or expert reading rather than a verified result. Audits make the gap concrete. Luo et al. (2025) define four failure modes—inappropriate benchmark selection, data leakage, metric misuse and post-hoc selection bias—and find several of them in controlled experiments on two open-source systems; Beel et al. (2025) re-ran the AI Scientist and found keyword-level literature search, ideas wrongly classified as novel and execution failures; Ding et al. (2026) screened 125 works and coded 26 on seven audit dimensions, finding code release common but reproducibility-grade and claim-verification artifacts rare, the "verification gap". Most recently XScientist proposes a git-like protocol in which hypotheses, attempts, claims, reviews and hand-offs are typed content-addressed objects with immutable checkpoints, claim–evidence closure and authority-aware gates, while stressing that a passing integrity check is not scientific truth (Luo, 2026). It is the nearest conceptual neighbour to the protocol studied here, but it is an artifact and implementation, not an experiment. The AI Scientist-v2, Kosmos, the pitfalls audit, the survey and XScientist are preprints at the time of writing.

1.2. Agent benchmarks for scientific programming and the oracle problem

A parallel literature evaluates agents task by task against executable oracles. ScienceAgentBench extracts 102 tasks from 44 publications, validated by nine experts, with programs as outputs, execution-based metrics and cost tracking; the best agent solves 32.4% of tasks and o1-preview 42.2% at more than ten times the cost, and the authors call for assessing individual tasks before bold claims about end-to-end automation (Chen et al., 2024). DiscoveryBench formalises discovery as dataset, metadata and goal across 264 real and 903 synthetic tasks (Majumder et al., 2024); DSBench supplies 466 analysis and 74 modelling tasks with long contexts (Jing et al., 2024); MLAgentBench evaluates a ReAct agent on 13 experimentation tasks (Huang et al., 2023b); MLE-bench curates 75 Kaggle competitions with human baselines, where the best setup medals in 16.9%, and studies resource scaling and contamination (Chan et al., 2024). CORE-Bench asks agents to reproduce published results across 270 tasks from 90 papers (Siegel et al., 2024); PaperBench asks them to replicate 20 ICML papers against 8,316 author-co-developed rubric items graded by an LLM judge validated on a separate judge benchmark (Starace et al., 2025). BLADE grades analysis *decisions* against independent expert analyses, admitting multiple valid paths (Gu et al., 2024); SWE-bench grades 2,294 GitHub issues by tests (Jimenez et al., 2023).

The stream also criticises its own oracles. Task-setup and reward-design flaws—SWE-bench Verified's insufficient tests, TAU-bench counting empty responses as successes—mis-estimate

performance by up to 100% relative, and a checklist reduced overestimation on CVE-Bench by 33% (Zhu et al., 2025). Agent benchmarks focus on accuracy without cost, with inadequate holdouts and conflated developer needs (Kapoor et al., 2024); the HAL harness later ran 21,730 rollouts for about $40,000 (Kapoor et al., 2025). Contamination must be measured per benchmark (Sainz et al., 2023); long-horizon reasoning and instruction following are the main agent failure causes (Liu et al., 2023); and τ-bench shows success and reliability over repeated trials are distinct quantities, with gpt-4o under 50% success and pass^8 below 25% in retail (Yao et al., 2024).

The evaluator itself is a threat. LLM judges show position, verbosity and self-enhancement biases even where strong judges agree with humans more than 80% of the time (Zheng et al., 2023); only the largest of thirteen judge models reach reasonable alignment, remain behind inter-human agreement and can differ from human scores by up to five points (Thakur et al., 2024); self-recognition correlates linearly with self-preference, so a model grading its own output is a biased oracle (Panickssery et al., 2024); and a simulated research-and-review loop with human annotation shows pronounced misalignment between LLM reviewers and humans (Li et al., 2025). Hence oracles here are deterministic, held on a host the executor cannot reach, and never the executor's own model grading itself. The checklist, HAL and the review simulation are preprints.

1.3. Workflows, provenance and reproducibility

Workflow engines locate reproducibility in re-executable, environment-pinned definitions: Snakemake's rules scale from workstation to cluster unchanged (Köster and Rahmann, 2012), and Nextflow combines dataflow with containers for cross-platform reproducibility (Di Tommaso et al., 2017). W3C PROV supplies the entity–activity–agent vocabulary (Missier et al., 2013), and ReproZip shows execution provenance can be captured automatically rather than self-reported (Chirigati et al., 2016). The norms are codified: track how every result was produced, avoid manual manipulation, archive program versions and seeds, publish scripts and results (Sandve et al., 2013); reproducibility as the minimum standard when replication is impossible (Peng, 2011); the consensus definitions of reproducibility (same data, code, conditions) and replicability (new data) (National Academies of Sciences, Engineering, and Medicine, 2019); and transparency, re-executability and determinism distilled from seven reproductions (Krafczyk et al., 2021).

The empirical record explains why norms must run rather than be declared. Of 204 *Science* papers under a data-and-code-on-request policy, artifacts were obtained for 44% and findings reproduced for 26% (Stodden et al., 2018); of more than 9,000 R files from replication datasets, 74% failed initially and 56% after automatic cleaning (Trisovic et al., 2022). Policy without executable verification has a low yield; the protocol arms therefore turn provenance requirements into gates that run.

1.4. Preregistration and specification discipline

Preregistration separates prediction from postdiction by fixing question and analysis plan before outcomes are observed (Nosek et al., 2018); Registered Reports review and provisionally accept the protocol before data collection (Chambers, 2013), and early evidence indicates they work without being a universal cure (Chambers and Tzavella, 2021). Flexibility in collection, analysis and reporting dramatically inflates false positives (Simmons et al., 2011); data-dependent choices do so even without intent, the garden of forking paths (Gelman and Loken, 2014); and hypotheses can be presented as a priori after the results are known (Kerr, 1998). The TOP guidelines and the reproducibility manifesto translate these concerns into graded standards and measures (Nosek et al., 2015; Munafò et al., 2017). The clearest quantitative effect is that the first hypothesis was positive in 96% of 152 standard reports but 44% of 71 Registered Reports (Scheel et al., 2021), against a background in which 100 replications recovered roughly one-third to one-half of original findings (Open Science Collaboration, 2015).

An agent under an open instruction walks the garden of forking paths unobserved; the structured-protocol arm is, functionally, a preregistration imposed on the agent, with phases, ledgers and stopping rules fixed before execution. The analogy is inexact: Registered Report authors self-select; arms here are assigned.

1.5. Human–AI delegation, automation bias and self-reported efficiency

Human-factors research supplies the vocabulary: use, misuse, disuse and abuse of automation (Parasuraman and Riley, 1997); automation of acquisition, analysis, decision selection and implementation along a manual-to-automatic continuum (Parasuraman et al., 2000); and the irony that the operator is left with what could not be automated and must monitor a system that rarely needs intervention (Bainbridge, 1983). Automation bias across 74 studies is mediated by cognitive style, system design and experience (Goddard et al., 2012); explanations raise acceptance regardless of correctness (Bansal et al., 2021); cognitive forcing reduced overreliance among 199 participants (Buçinca et al., 2021), the principle behind having reviewers execute checks rather than read summaries. People prefer machine-in-the-loop designs to full AI control, with trust the dominant factor (Lubars and Tan, 2019); the design space of such studies is surveyed by Lai et al. (2023); and a preregistered meta-analysis of 106 experiments finds human–AI combinations on average below the best of either alone, with gains only in content creation (Vaccaro et al., 2024).

The decisive result for measurement is a randomised trial of 16 experienced developers on 246 tasks: AI tools *increased* completion time by 19%, although developers forecast a 24% speed-up and still believed in a 20% speed-up afterwards (Becker et al., 2025). Perceived and measured efficiency diverged in sign, which is why the cost of delegation here is read from logged time and

resource use, never from self-report. The trial is a preprint; the methodological point stands.

1.6. Durable execution and resource accounting

Long autonomous runs fail like distributed systems. Most outages stem from software and operations rather than hardware, motivating process pairs and transactions (Gray, 1986); tail-tolerant techniques absorb latency variability (Dean and Barroso, 2013); retries are safe only under idempotence (Helland, 2012); chaos engineering verifies reliability by experimenting with failure (Basiri et al., 2016); sampled tracing makes behaviour observable (Sigelman et al., 2010); and durable workflows with persistent state have a formal semantics (Burckhardt et al., 2021). Agent-era work adds a taxonomy of artifacts to trace (Dong et al., 2024), aligned checkpoints for rewind and resume (Zhuang et al., 2026), evidence over 23,392 episodes that capability and reliability diverge as tasks lengthen (Khanal et al., 2026), and cost-of-pass, the expected cost of a correct solution, as the economic unit (Erol et al., 2025); these four are preprints. Orchestrators such as Temporal, Airflow and Argo embody the same principles as software. This study borrows the apparatus—idempotent checkpointed steps, traced runs, cost per correct outcome—and treats recovery after failure as an outcome of the protocol.

1.7. Claim verification and fabricated references

FEVER labels 185,445 claims Supported, Refuted or NotEnoughInfo with Fleiss kappa 0.68 (Thorne et al., 2018); SciFact pairs 1.4K expert claims with evidence and rationales (Wadden et al., 2020); Attributed QA requires generation to cite its support (Bohnet et al., 2022); RARR attributes and post-edits any generator's output (Gao et al., 2023); FActScore scores atomic facts, finding only 58% of ChatGPT biographical facts supported (Min et al., 2023). Self-consistency improves reasoning by sampling and agreement (Wang et al., 2022), but models struggle to self-correct without external feedback and can degrade when they try (Huang et al., 2023a); the gates here are external for that reason. Fabricated references are the most visible failure: 55% of GPT-3.5 and 18% of GPT-4 citations in 84 short reviews were fabricated (Walters and Wilder, 2023), as clinical case reports had documented early (Alkaissi and McFarlane, 2023); nearly 300 ACL, NAACL and EMNLP papers of 2024–2025 contain a hallucinated citation and more than 100 were accepted (Sakai et al., 2026); and an audit of 111 million references conservatively estimates 146,932 hallucinated citations in 2025 (Zhao et al., 2026). The first study used 2023-era models and the last is a preprint with an unknown false-negative rate; both motivate a gate that resolves every reference against a primary metadata record.

1.8. Positioning

Before naming the design we searched for prior use. An exact-phrase search (web and arXiv titles, 2026-10-29/30) for "protocol-guided autonomous research" returned no paper, system or project using it. The nearest hits are XScientist, where "protocol" denotes a git-like provenance and hand-off standard (Luo, 2026), and ProtoPilot, where it denotes a wet-laboratory procedure for robots; neither uses the phrase and neither compares instruction structures. The acronym "PGAR" returned no match in the AI or science-agent context; it is used elsewhere for unrelated entities such as gene and company names.

Table 1 places the closest designs beside this study. Systems hold their pipeline fixed, vary human involvement or propose a protocol without an experiment; benchmarks hold the task fixed and vary the model or scaffold; audits inspect traces after the fact. None varies the instruction regime while holding the executor constant and grading with oracles the executor cannot see.

Prior work has established that end-to-end pipelines produce reviewable papers cheaply, that staging plus human feedback helps, that traceable evidence chains can be built in, that such systems commit leakage, metric misuse, post-hoc selection and citation errors visible only in traces, that executable per-task oracles with cost accounting exist, and that a git-like research protocol has been proposed. What this study adds, and should claim no more than, is a controlled four-arm comparison in which the executor—Claude Code 2.1.314 in bare mode, model claude-opus-5-5, isolated on cluster nodes—is held fixed and the instruction regime varied: open instruction (A); a structured protocol with phases, ledgers and stopping and recovery rules (B); the same with controller-enforced executable gates (C); and the same with a separate review-and-repair session (D), across six task families from data audit to integrated mini-projects, with hidden deterministic oracles on a separate host and outcomes defined as scientific validity rather than reviewer score. The novelty is the experimental design and its measurement, not the idea that protocols help, which the systems above already presuppose.

2. Conceptual framework: delegation contracts, autonomy and evidence

2.1. What is delegated when research is delegated

When a researcher hands a scientific task to an autonomous agent, three things change hands: a description of what is wanted, a bundle of resources with which to obtain it, and the authority to decide, along the way, among the many admissible choices that any empirical analysis entails. The first two are explicit. The third is the source of most of the trouble, because the choices an agent makes on its own (which observations to drop, which standard errors to report, whether information was available when it was used, when to declare the work finished) are precisely the

Table 1. Closest prior systems, benchmarks and audits, and the design of this study. Cells summarise the sources cited in the text; "n/a" means not reported in the source consulted.

System	Inputs / workflow	Verification mechanism	Human involvement	Resources reported	What is compared
AI Scientist (Lu et al., 2024)	Code template; fixed idea-to-paper pipeline	In-system LLM reviewer; code runs	None during run	Under $15 per paper	Nothing; single pipeline
AI Scientist-v2 (Yamada et al., 2025)	Topic prompt; agentic tree search	In-system reviewer; one workshop acceptance	Minimal	n/a	Nothing; single pipeline
Agent Laboratory (Schmidgall et al., 2025)	Human idea; three staged solvers	Human survey; automated review scores	Optional feedback each stage	84% lower expenses	Copilot vs autonomous; models
Coscientist (Boiko et al., 2023)	NL goal; tools incl. cloud lab	Physical outcome; author inspection	Supervise execution	n/a	Nothing; six tasks
Co-Scientist (Gottweis et al., 2026)	Research goal; tournament loop	In-system critique; wet-lab downstream	Scientists specify, validate	n/a	Test-time compute scaling
data-to-paper (Ifargan et al., 2025)	Annotated data; stepwise agents	Back-traced numbers; human verification	Optional copilot steps	n/a	Autopilot vs copilot
Kosmos (Mitchener et al., 2025)	Objective + dataset; world model	Expert reading of reports	None during run	Up to 12 h, 200 rollouts	Nothing; case studies
XScientist (Luo, 2026)	Research state as typed objects	Claim–evidence closure; integrity gates	Fork, inspect, extend	n/a	No experiment reported
ScienceAgent-Bench (Chen et al., 2024)	Task + data; program output	Execution metrics; figure judging	Experts validate tasks	API cost per task	Models and frameworks
PaperBench (Starace et al., 2025)	Paper text; replicate from scratch	Author rubrics; validated LLM judge	Rubric co-development	In paper	Models and scaffolds
BLADE (Gu et al., 2024)	Dataset + question	Decisions vs expert analyses	Experts give ground truth	n/a	Models and prompting
Pitfalls audits (Luo et al., 2025; Beel et al., 2025)	Existing open-source systems	Auditors inspect traces, code	Auditors only	n/a	Failure modes present or not
This study (WP29)	Fixed task; one of four instruction regimes	Hidden oracles on separate host; executable gates	None in loop; interventions ledgered	Logged time and cost per run	Four arms, executor fixed

ones that determine whether the delivered artefact is scientifically valid. The "garden of forking paths" identified for human analysts (Gelman and Loken, 2014) does not disappear when the analyst is a language model; it is walked faster.

We therefore describe a delegation as a *contract* rather than a prompt. A contract fixes, before execution, the questions and estimands, the inputs and their availability restrictions, the authorised actions, the ordering of phases, the verification criteria, the resource budget, the rules for recovery and amendment, and the evidence the executor must leave behind. Formally, a protocol is the tuple

$$\mathscr{P} = (Q, \Theta, \mathscr{D}, \mathscr{A}, \mathscr{G}, \mathscr{V}, \mathscr{B}, \mathscr{R}, \mathscr{L}), \tag{1}$$

whose components are listed in Table 2 together with the concrete file or mechanism that serialises each of them in our implementation. We use the notation only as an index: it is not a theorem and it establishes nothing by itself. Its value is that every component has a place where it is written down, hashed and, where possible, checked by a machine rather than by the executor.

Table 2. Components of the delegation contract and their serialisation in the reference implementation.

Component	Meaning	Serialisation and consumer
Q	scientific question and scope	`TASK.md` (question section); read by executor and evaluator
Θ	estimands, hypotheses, admissible interpretations	`TASK.md` (estimand and availability rules); hidden `REFERENCE.md` on the evaluator host
$\mathscr{D}$	inputs, versions, information availability	`data/` with `README.md`, `bundle.json` (SHA-256 of every file); verified on the node before launch
$\mathscr{A}$	authorised actions and tools	launcher flags: bare mode, fixed tool inventory, no network, no sub-agents; identical across arms
$\mathscr{G}$	phase dependency graph	protocol text (arms B–D) and the executor-maintained ledger `out/state/phases.json`; harness run-state machine
$\mathscr{V}$	verification criteria	`gate_spec.json` and the gate program (oracle-free, runtime); `evaluate.py` with `oracle.json` (hidden, final)
$\mathscr{B}$	budgets and stopping	`configs/budgets.yaml`: USD cap, wall clock, turns, gate and review cycles; enforced by the client cap and a remote kill
$\mathscr{R}$	recovery and amendment	protocol recovery rules; session resumption; attempt-numbered reruns; `protocol/amendments.csv`
$\mathscr{L}$	evidence and provenance	SQLite state store (runs, transitions, sessions, events, receipts, reviews, screenshots, faults, jobs, interventions) and per-run directories

2.2. Phase states and enforceable invariants

Within a contract, a phase is in one of seven states: *pending*, *running*, *passed*, *failed*, *blocked*, *stale* or *skipped_optional*. A phase becomes eligible only when its required predecessors are *passed*; an artefact produced by a phase is valid only relative to the versions of inputs, code, configuration and criteria used to produce and check it, so that a change upstream makes downstream results *stale* rather than silently wrong. At the level of the whole run, the harness keeps a second, smaller state machine (*not_started*, *prepared*, *running*, *completed*, *timeout*, *budget_exhausted*, *error*, *interrupted*, *evaluated*, *void*) whose transitions are validated in code.

Five invariants are enforced mechanically. (1) A run cannot be marked *evaluated* without an evaluation receipt. (2) A changed upstream input invalidates downstream results, because the gate regenerates the results from the replication script and compares them. (3) A run cannot consume more than the envelope: each session is launched with a cap equal to the remaining budget, and the controller kills the process at the wall limit. (4) A claim cannot be marked supported unless it references result keys that exist, and the final evaluator scores claims against protected references. (5) A skipped optional phase cannot support a claim, because the final evaluator ignores phase ledgers and scores artefacts only. We state the limits of each checker in Section 4: these are workflow guarantees, not guarantees of truth. A gate that passes tells the researcher that the deliverable has the promised shape, reproduces from its own script and is internally consistent; whether it is *right* is a separate question answered by an evaluator the executor never sees.

2.3. Observable autonomy

The word "autonomous" conflates two outcomes that must be kept apart. *Workflow autonomy* means that the phases were executed without operational human guidance; *scientific completion* means that the artefacts satisfy the independent task rubric. The cell of interest is *intervention-free verified completion*, where both hold. Its complements are informative: *autonomous execution with incorrect output* (no intervention, scientific failure) is the failure mode that audits of autonomous research systems describe (Luo et al., 2025; Beel et al., 2025), and *correct assisted completion* (success after a substantive human intervention) is what a laboratory usually obtains in practice. The implementation records the four cells separately: the run status and terminal reason capture workflow autonomy; the evaluation receipt captures scientific completion; the intervention ledger (empty for every executor session in this study, because no human touched a running session) captures assistance. Every table and figure in the paper reports the primary endpoint as the conjunction and never relabels an assisted or budget-exceeding success as an autonomous one.

2.4. Why a controlled comparison

Published autonomous research systems typically present a single pipeline whose quality is judged by reviewer scores or by a few flagship outputs (Lu et al., 2024; Yamada et al., 2025; Gottweis et al., 2026). Such designs cannot say which element of the pipeline (the staging, the self-review, the evidence records, the sheer length of the instructions) produced the outcome, nor whether the outcome is scientifically valid rather than persuasive. The present study holds the executor fixed (same client, same model, same tools, same inputs, same envelope) and varies only the organisation and checking of the work across four arms. The contrasts between adjacent arms isolate, as far as the platform permits, the marginal contribution of structure (B−A), of executable gates with controlled state transitions (C−B) and of a separate review-and-repair stage (D−C), while D−A measures the whole method against an information-matched open instruction. We do not claim that this identifies a "pure effect of structure": the protocol text is longer than the open instruction, gates return feedback and reviews consume part of the budget. A length-matched unstructured control addresses the first confound; the other two are properties of the treatments and are reported as costs rather than hidden.

3. Corpus-informed method design

The protocol compared in arms B–D was derived from a content coding of the instruction files—the `CLAUDE.md` mandates—that the author had previously issued to Claude Code for earlier working papers in this series. The aim was to learn what a researcher writing such mandates repeatedly asks for and omits, and to turn the omissions into design elements. The exercise motivates the design; it makes no prevalence claim.

3.1. Corpus and coding

Thirteen files were retrieved from the author's git server on 2026-10-29 and hashed (SHA-256) into a manifest against which every file was re-verified before coding. They contain eleven distinct mandates (WP15, WP18–WP26 and WP28), one supplementary design brief (WP27), and one byte-identical duplicate of the WP15 mandate, which is recorded but counted once. The WP27 brief is a post-implementation description of a simulator rather than an instruction file and is excluded from every "of 11" count; WP27's own mandate could not be retrieved (HTTP 404) and nothing is inferred about it. The mandates range from 1,092 to 8,809 words (median 1,966); two (WP15 and WP28) are an order of magnitude longer than the rest, the extra length tracking engineering prescription rather than substantive research rules.

Coding followed a written codebook. Automated variables (length, headings, phase and gate

labels, keyword families) were computed by a script from the file bytes; analyst variables were assigned in a single automated-plus-LLM-analyst pass after reading each document in full, each judgement recorded as *yes*, *partial* or *no* with a verbatim quotation in the same cell so that it can be re-checked against the source. *Partial* means the feature is addressed only indirectly, conditionally or without an operational mechanism. There was no human second coder, so no intercoder reliability is reported and none should be inferred. Table 3 gives the counts.

3.2. What the mandates consistently require

Eight patterns recur. (1) Every mandate opens by casting Claude Code as a multidisciplinary research team and immediately prohibits fabrication of data, results, citations, proofs or significance (11 of 11), the most stable element of the corpus. (2) Ten declare ordered phases (five to thirteen; median seven), nine make drafting conditional on validated results, and ten name gates, mostly as numbered acceptance gates. (3) All eleven require a verified literature review or novelty gate before any contribution claim. (4) Ten include audit or self-review as a phase: claim-to-file audits, an adversarial referee report, three simulated referees, or a red-team agent. (5) Nine require traceability of every number to a result file, registry or hashed manifest. (6) All six data-based mandates impose information-set discipline (look-ahead, as-of-date, vintage and chronological-split rules). (7) Eight name baselines. (8) Most provide an honest-failure fallback when work is blocked.

Four excerpts show the register. The integrity preamble: "Never fabricate data, results, citations, institutional details, variable definitions, statistical significance, sample sizes, or conclusions. Never write a result in the paper before it has been generated and verified from a saved machine-readable output." (`WP15_CLAUDE.md`). A phase gate before drafting: "do not start manuscript drafting until the baseline model is validated, claimed GE equilibria clear markets, principal simulations and exhibits reproduce, bibliography is verified, and theoretical claims have proofs or appropriate qualifications. If blocked, document the issue honestly and deliver a scoped partial result." (`WP18_CLAUDE.md`). A pre-specified primary outcome: "Pre-register primary endpoints and hypotheses before running final test evaluations. Primary outcome: out-of-time absolute log-price error on arm's-length residential transactions." (`WP26_CLAUDE.md`). A declared acceptable null: "A dead end with a mathematical explanation is better than a fabricated breakthrough. A strong negative result or counterexample may itself become the contribution." (`WP28_CLAUDE.md`).

3.3. What the mandates consistently lack

The same documents are silent on mechanism. (1) Gates are declarative. Only WP15 and WP28 attach a gate to something that can fail mechanically—"The release script must fail if: a table references a missing result; a figure is missing source data; LaTeX does not compile; citations

Table 3. Feature counts over the eleven distinct mandates (WP27 brief and the WP15 duplicate excluded). "Partial" is defined in the codebook as present but indirect, conditional or without an operational mechanism; "—" means the category was not used for that feature.

Feature	Yes	Partial	No	Note
Explicit research question (verbatim)	10	—	1	WP22 states objectives, not a question
Explicit estimand or hypotheses	7	4	0	H1–H6 lists, theorem queues, primary outcome
Assumptions stated in the mandate	2	8	1	most require assumptions without stating them
Falsification / null-result conditions	6	4	1	
Literature / novelty requirement	11	0	0	universal; usually a "novelty gate"
Data-access controls	7	4	0	credentials, scraping, privacy, licensing
Temporal-information controls	6	1	4	all six data-based mandates; n/a for theory
Baseline / benchmark requirement	8	3	0	
Phases declared	10	—	1	5 to 13 phases; median 7
Dependency / gate structure	10	1	0	"partial" = implicit ordering (WP21)
Stopping rules	2	6	3	"partial" = document and deliver partial result
Recovery / resume instructions	2	6	3	"partial" = progress logs, caching, retry limits
Provenance requirements	9	2	0	manifests with hashes in WP15, WP28
Review / self-audit stage	10	1	0	results audits, simulated referees, red team
AI-use / disclosure statement	1	0	10	WP28 only
Budget / resource limits	1	3	7	WP26 asks for API-cost tiers; no numeric cap
Integrity prohibitions ("never invent …")	11	0	0	universal, usually in the opening paragraph
Pre-registration mentioned (keyword)	4	—	7	no registry or timestamp mechanism
Random seeds mentioned (keyword)	11	—	0	
Statements marked as added or reconstructed	6	—	5	

are unresolved; . . . " (`WP15_CLAUDE.md`)—and no mandate specifies who or what checks a gate other than the executing agent. (2) Verification is not independent. "Independent audit", "independently checked", "independently verified", "hostile-referee critique" and "three independent review perspectives" are all to be performed by the same agent; WP28 alone names the tension: "Record unresolved issues candidly; do not have the same model merely declare its own work accepted." (`WP28_CLAUDE.md`). (3) Pre-registration is not operational: four mandates ask for it, none specifies a registry, a timestamp, a hash or a freeze. (4) Stopping rules are rare: only WP15, with a dedicated stop-conditions section, and WP28 define when to stop; most others say to continue with feasible work. (5) Recovery is rare: only WP15 (resumable, idempotent, checkpointed jobs) and WP25 require it. (6) Budgets are absent: seven of eleven set no resource limit, only WP26 asks for an API-cost budget, and no mandate states a numeric token, dollar or compute cap. (7) Ten of eleven say nothing about disclosing AI involvement in the paper. (8) Success criteria are not quantified: targets read "substantially improve", "publication-quality" or "the standard expected in a top finance journal", none checkable from the document. (9) The instruction text itself has uncertain provenance: WP23 states that part of its protocol was reconstructed by an assistant after truncation, WP24 and WP26 carry sections marked as added on 2026-10-27, and WP18, WP20 and WP25 carry prefatory repository-mapping notes, so the mandate as retrieved is not always the mandate as issued.

3.4. What the corpus cannot establish

These documents are instructions: they record what the author asked Claude Code to do, not what was done. The coding therefore provides no evidence about how the earlier working papers were actually executed, whether the instructions were followed, or whether the papers were produced without human assistance; several mandates in fact anticipate human steps such as expert raters and provision of credentials. A mandate that requires a provenance manifest is coded as requiring one; whether the manifest exists is outside the coding. The corpus is also small and researcher-selected—eleven files by one author for one series—and was coded in a single pass without reliability statistics. Its counts motivate design patterns; they do not estimate the prevalence of any practice among autonomous-research mandates in general, and no inferential statistic is computed from them. Questions about execution require run-level evidence, which the experiment that follows is designed to produce.

3.5. From absences to design elements

The corpus shows a stable rhetorical protocol (role, integrity rules, phases, gates, audits, traceability) and an equally stable absence of mechanisms. Rather than add prose rules, arms B–D turn each

absence into something checkable. Uncertain provenance of the instruction (absence 9) and non-operational pre-registration (absence 3) become a *hashed, frozen mandate*: the instruction injected into each run is fixed and hashed before execution, so that what the executor received is known exactly. Declarative gates (absence 1) become *executable gates* enforced by the controller in arms C and D, which can fail mechanically rather than by the agent's own judgement. Self-performed audits (absence 2) become *external evaluators*: hidden deterministic oracles on a separate host, and in arm D a review-and-repair session separate from the executing session, so that no model merely declares its own work accepted. Missing budgets and stopping rules (absences 4 and 6) become *explicit budgets and stop rules* in the protocol. The unrecorded human steps that several mandates anticipate become an *intervention ledger* in which any human intervention during a run is recorded. And the audit phase that the mandates ask the same agent to perform becomes the *separate review* that distinguishes arm D from arm C. Arm A, the open instruction, receives the task without any of these mechanisms, and so serves as the control against which their contribution is measured.

4. PGAR implementation

The reference implementation is a small Python harness (about 1,300 lines in `src/pgar/`, standard library plus `numpy`, `pandas`, `pyyaml` and `jsonschema` on the controller; the gate program is standard-library only so that it runs with any interpreter on an executor node). Figure 1 shows the components and the trust boundaries between them. We describe the roles, the isolation mechanism, the launcher, the gate, the review stage, the state store and the capture subsystem, and we state explicitly what each mechanism does not do.

4.1. Roles and trust

Six roles are distinguished. The *builder* is the Claude Code session that developed this study (model `claude-fable-5-1`, client 2.1.314) under the researcher's mandate; it wrote the harness, the protocols and the smoke task, and it delegated the authoring of the task suite and the literature review to fresh sub-sessions. The *executors* are fresh Claude Code sessions (client 2.1.314, model `claude-opus-5-5`) launched in bare mode on four cluster nodes. The *runtime verifier* is the gate program, a deterministic script without reference answers. The *final evaluator* is a deterministic task-specific script executed only on the evaluator host against a hidden oracle. The *reviewer* (arm D only) is a separate Claude Code session with its own system prompt. The *reporting pipeline* consists of the export, analysis, table and figure scripts that produce every number in this paper from the state store.

The builder's involvement in task design is a limitation that no amount of session freshness removes: the same organisation (the researcher's Claude Code account, two model identities from

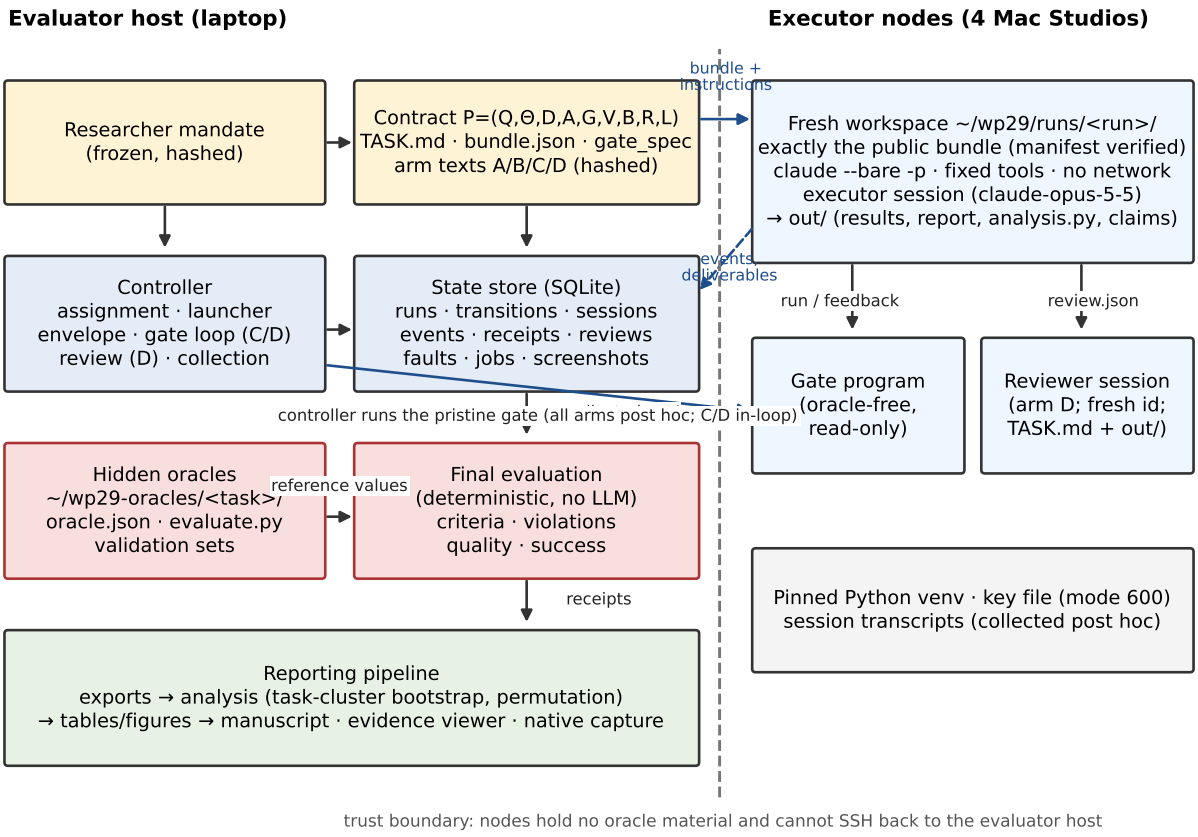


Figure 1. Architecture of the reference implementation (diagram, not a screenshot). The controller, the state store, the oracles and the evaluators live on the evaluator host; executor sessions run on separate cluster nodes in fresh workspaces that contain only the public task bundle. Arrows crossing the dashed boundary carry only the bundle, the instructions, the stream of events and the deliverables; reference answers never cross it.

the same vendor) authored the tasks, the oracles and the executors' instructions. We mitigate it in the ways that are available (reference values from analytical identities, independently written code or source documents with locators; validation of every evaluator against seeded correct and incorrect artefacts; a frozen interface written before any task), and we report, for every task, who could access the reference answers at each stage.

4.2. Isolation

Three mechanisms separate executors from hidden material. First, a physical boundary: executors run on cluster nodes that hold no copy of the repository, the oracles or the earlier working papers, and the nodes cannot open an SSH connection to the evaluator host (verified by a port probe). Second, a workspace boundary: each run receives a fresh directory containing exactly the public bundle, verified before launch by comparing a SHA-256 manifest computed on the node with the local one; a mismatch aborts the run as an infrastructure error. Third, an instruction boundary: executors are started with `-bare`, which the client documents as skipping `CLAUDE.md` discovery, hooks, plugins, MCP servers, auto-memory and skills. We did not take the documentation on trust. Sentinel probes with an innocuous project `CLAUDE.md` showed that an ordinary print-mode session on the laptop loaded about 49,000 tokens of context, including the researcher's cluster instructions, while a bare session loaded 144 tokens and reported no sentinel; the same probe on each executor node reported no sentinel and no cluster instructions with 175–177 input tokens (Section 5). The treatment text is injected explicitly with `-append-system-prompt-file`, and every run records a hash of the effective context (common preamble, arm text, task statement and user prompt).

What isolation does not do: executors share a user account on each node and could, with the shell tool, read sibling workspaces or the transcript of another session; the gate program is world-readable on the node and arms A and B were not told about it. We do not prevent these behaviours; we *audit* them by scanning every tool invocation in the verbatim event stream for paths outside the workspace and for invocations of the gate, and we report the counts per run.

4.3. Launcher and resource envelope

A session is launched over SSH as `claude -bare -p` with a pre-generated session identifier, structured streaming output, a fixed tool inventory (`Bash`, `Read`, `Write`, `Edit`, `Glob`, `Grep`; no web access and no sub-agents), the listed tools pre-approved and every other permission request denied, a model identifier, a budget cap and a turn limit. The job is written to a ledger *before* submission, which is what makes duplicate submissions detectable. Each line of the output stream is appended verbatim to a raw file and normalised into an event record (type, tool, status, usage, hash of the payload) as it arrives. If the wall-clock limit is reached, the controller kills the remote process

by its unique session identifier and records the run as a timeout; the artefacts produced so far are still collected and evaluated. A session can be resumed with the client's `-resume` flag, which the harness uses for gate feedback (arms C and D), review feedback (D) and the recovery experiment. The envelope is shared: a continuation is launched with a cap equal to the remaining budget, and reviewer sessions draw from the same envelope as the executor they review. The client reports costs as list-price estimates, which can slightly exceed the cap; the harness records both the cap and the observed estimate and never substitutes an invented figure. No temperature or seed control exists in the client; repeats are therefore genuine fresh sessions.

4.4. The gate

The gate program checks the output contract of a task without knowing the answer. Given the public `gate_spec.json`, it verifies that the required files exist, that `results.json` parses and contains every required key with the declared type and nullability and finite numbers, that the report has the six required sections, that the claims ledger is a non-empty list whose entries reference existing result keys, that an exclusion ledger (if present) sums correctly, that a phase ledger (if present) has no phase marked *passed* without recorded checks and no non-terminal phase, that confidence intervals contain their estimates, that standard errors are positive, that declared date pairs respect chronology and, finally, that the replication script regenerates `results.json` in a clean copy within a relative tolerance of 10^{-6}. It writes a receipt with every check and its detail. In arms C and D the executor is told to run the gate before declaring completion, and the controller runs its own pristine copy after the session ends; a failing receipt is returned to the executor as a resumed turn, at most three times, within the envelope. In arms A and B the same program runs once after the session, post hoc and silently, to provide the common contract check and the fresh re-run reproducibility measure for all arms.

4.5. Separate review

In arm D, once the controller gate passes, a new session is started in the same workspace with a reviewer system prompt. The reviewer is told that it has not seen how the deliverables were produced and must not look for the transcript; it may re-run code; it has no reference answers; it must write a structured verdict (*accept* or *revise*) with located issues. If the verdict is *revise*, the executor session is resumed once with the review, after which the gate runs again. The review is "separate" in the sense of a fresh context and a different instruction, not in the sense of an independent judgment: it is the same model family as the executor, and we make no claim of independence (Panickssery et al., 2024).

4.6. State store, evidence and capture

All state lives in a single SQLite database opened in write-ahead-logging mode with explicit immediate transactions and serialised access from the controller's worker threads. Eleven tables record runs and their validated transitions, sessions (with the exact command, the cost reported by the client and the first-turn input size), events, gate receipts, reviews, evaluations, screenshots, faults, jobs and interventions. Analysis-ready CSV exports are regenerated from the database by a single script, and a read-only HTTP viewer renders the same rows for inspection and for dashboard captures. Screenshots are taken with the operating system's native capture utility on a window identified through the window server, never synthesised; each capture is logged with its surface, mode, timestamp, hash and the run or event it documents.

5. Experimental design

5.1. Tasks

The benchmark comprises 24 confirmatory tasks, four in each of six families, plus seven development tasks (one smoke task and one development task per family) used only for the smoke test and the pilot. Table **??** lists every confirmatory task with its data type, reference-answer provenance, planted condition and difficulty rationale; the task cards in Appendix **??** give the full statements. The families follow the mandate: F1 data provenance and audit (planted duplicates, unit mixes, vintage misalignments, a changing country universe, a final-vintage variable unavailable at origin, a corrupted file reconciled to a control total); F2 estimation and inference (two-way fixed effects, design-based uncertainty under clustering, a memo with an inappropriate i.i.d. standard error under serial dependence, Monte Carlo coverage, Okun's law on FRED data); F3 forecasting and temporal information (rolling-origin evaluation on public series with release lags, real-time vintages, a future-informed normalisation trap and overlapping multi-horizon targets); F4 source-grounded evidence synthesis (coefficient extraction with locators, assertion checking, harmonisation of incompatible effect-size definitions, citation verification, all on CC BY arXiv e-prints and a Statistics Canada release); F5 theory and numerical validation (a Cobb–Douglas exchange economy with a broken aggregate identity, the Marriott–Pope AR(1) bias, a false proposition requiring a counterexample, value-function iteration against a closed form); F6 integrated mini-projects (a Preston curve, a genuine null result on weekday effects in Treasury yields, an invalid-premise event study, Okun's law with a stability test, a planted confounder). Of the 24 confirmatory tasks, 14 use observed public data (FRED, ALFRED, World Bank, Bank of Canada), 5 are source-document tasks, and 7 have a null, infeasible or false-premise truth, so that honest recognition rather than completion is the correct answer.

Every task is delivered as an immutable bundle (statement, inputs with provenance, a public oracle-free gate specification, a manifest with the SHA-256 of every file) and has a hidden oracle (reference values, tolerances, required criteria, critical-violation checks, a deterministic evaluator and a seeded validation set). Reference values come from analytical identities, from code written independently from any solution, or from source documents with table and page locators; no reference was produced by a language model solving the task. Every evaluator was validated before the pilot against at least one independently written correct solution and at least two wrong ones (Appendix **??**); the pilot then exposed one harness defect in how evaluators were invoked (Section **??**), which was corrected before the confirmatory stage.

5.2. Treatments

Table **??** summarises the four arms. All arms receive the same task statement, inputs, output obligations (a results file with named keys, a six-section report, a replication script that regenerates the results from the inputs, and a claims ledger), the same common preamble with environment facts and integrity rules, the same client, model, tool inventory, permission policy, offline node, pinned Python environment and resource envelope (4.00 USD, 1,500 s of wall clock, 80 turns per session). The arms differ only in the text appended to the system prompt and in what the controller does after a session ends. Arm A appends one sentence. Arm B appends a six-phase protocol (scope and contract; data audit with an exclusion ledger; baseline; method and replication script; verification; reporting with a claims ledger) with stopping and recovery rules and a phase ledger. Arm C adds a mandatory, executable gate: the executor must run the gate program before declaring completion, and the controller re-runs its own copy after the session and returns failures as resumed turns, at most three times within the envelope. Arm D adds a separate review-and-repair session after the gate passes, drawn from the same envelope. The appended texts are 6, 665, 827 and 921 words long (common preamble 279 words); the first assistant turn of the smoke runs consumed 2,238, 3,717, 4,111 and 4,319 input tokens respectively, the difference being the protocol text only. A length-matched unstructured control (general research advice, 533 words, no phases, ledgers or gates) is used in the ablation of Section **??**.

5.3. Assignment, blocking and repeats

The design is paired: every confirmatory task is run under every arm with 3 repeats, for 0 assigned runs. A seeded generator shuffles the order of task blocks and, within each block, the order of the (arm, repeat) cells; runs are dispatched to four executor nodes with two concurrent sessions per node, so that the arms of a task block execute interleaved in time and no arm is systematically earlier than another. Repeats are fresh sessions in fresh workspaces; the client exposes no seed or

temperature control, so model stochasticity is not controlled and repeat outcomes are preserved as observed. The assignment manifest and every run's node, order and timestamps are stored before launch.

5.4. Endpoints and estimands

The primary endpoint for run (i,a,r) is

$$Y_{iar} = \mathbf{1}\{\text{required criteria satisfied} \wedge \text{no critical violation} \wedge \text{completed within the envelope} \wedge \text{no human intervention} \tag{2}$$

and the task-weighted completion rate of arm a and the paired contrast are

$$\hat{p}_a = \frac{1}{J}\sum_{j=1}^{J}\frac{1}{R_j}\sum_{r=1}^{R_j} Y_{jar}, \qquad \hat{\Delta}_{ab} = \hat{p}_a - \hat{p}_b. \tag{3}$$

The primary contrast is D−A; B−A, C−B and D−C are prespecified mechanism contrasts with Holm adjustment. Uncertainty uses a task-cluster bootstrap (5,000 resamples of tasks, percentile intervals of the paired difference) and a sign-flip permutation test on the task-level paired differences (20,000 permutations); with $J = 24$ task clusters, intervals are wide, and we report them as such. Secondary endpoints are correct completion regardless of the envelope, quality components (correctness, integrity, reproducibility, reporting, baseline; frozen weights 0.40/0.20/0.15/0.15/0.10), critical violations by type (leakage, fabrication, invalid inference, unreported substitution, false completion, provenance), fresh-environment reproducibility, correct recognition of infeasible or null-truth tasks and false blocking on feasible ones, and cost, usage and wall time with full denominators. The analysis plan, the scale decision rule and the freeze procedure were written before the pilot (`protocol/preregistration.md`) and frozen with hashes before the confirmatory stage; every departure is an amendment in `protocol/amendments.csv`.

6. Evaluation

6.1. Protected oracles and deterministic scoring

After a run ends, the controller collects the deliverables, runs the common contract check (the gate program, post hoc for all arms, which also yields the fresh re-execution reproducibility measure), and then invokes the task's evaluator on a composed workspace that contains the public bundle and the collected deliverables and nothing else. Evaluators are deterministic Python programs; none calls a language model. Each returns a receipt listing every required criterion with the observed value, the reference and the tolerance, every critical violation with its evidence, and the five quality

components; `scientific_success` is true only if every criterion passes and no violation is present. Receipts are validated against a public JSON schema, hashed and stored. Tolerances were set by the task designers from statistical or numerical reasoning (sampling variability of the reference quantity, differences between valid conventions such as CR0/CR1 cluster adjustments, t versus normal critical values, Newey–West bandwidths, binomial Monte Carlo error) so that valid implementations pass and wrong science fails; each justification is recorded in the task's hidden `REFERENCE.md`.

Critical violations are detected from deliverables only. Fabrication is flagged when reported input counts do not match the files, when a reported value cannot be regenerated from reported components (a residual, a coverage count, a Kolmogorov–Smirnov statistic), or when a value said to be printed in a source does not appear there. Leakage is flagged in forecasting tasks when the executor's out-of-sample errors fall below an information-feasible bound or match a leaky reference implementation, and in the F1 task with a final-vintage variable when that variable enters the model. Invalid inference is flagged when a conclusion contradicts the reported interval, when a significance claim rests on a standard error that ignores the stated dependence, or when a claim in the ledger asserts a universal statement on the basis of simulation-only keys. Unreported substitution is flagged when a cleaned aggregate is reported without the corresponding exclusion-ledger step or data-issue entry. False completion is flagged when a feasible task is declared partial or infeasible, when an infeasible one is declared complete, or when a not-assessable item is marked supported. These rules are task-specific, deterministic and documented; they cannot see intent, and we report them as what they are.

6.2. Evaluator validation

Every oracle ships a validation set: one or more independently written correct solutions whose replication script is actually re-run, and two to six wrong solutions embodying the failure the task is designed to detect (a leaky model, a naive estimator, a fabricated count, a false declaration of completion, a silent substitution). Before the pilot, all 31 oracles accepted their correct artefacts and rejected their wrong ones (Appendix **??**). The seeded sets are small and designer-authored: they bound gross evaluator errors, not the space of valid solutions, and the pilot was used to look for false rejections of legitimate variation.

6.3. Blinding and model-based grading

Evaluators are programs and see no arm label; run identifiers are neutral. Artefacts can reveal the arm to a reader (a phase ledger exists only in arms B–D), which is irrelevant to deterministic scoring but would matter for any human or model-based review; we therefore do not describe the evaluation as blinded. No model-based grading enters the primary or secondary endpoints. The reviewer in

arm D is part of the treatment, not of the evaluation.

6.4. Human involvement

No human reviewer was available for the benchmark; cells that would require one are marked unavailable. No human intervened in any executor session. The human contributions to this study are the mandate itself (prepared outside the session; preparation time unknown and reported as unknown), the decision to run it, and the final review of the manuscript; they are logged in `reports/interventions.csv` with their measurement basis. The builder session's own activity (harness construction, task-suite delegation, analysis and writing) is recorded in the phase log and in the reflexive analysis of Section **??**; it is a cost of the method, not an intervention in the experiment.

References

Alkaissi, Hussam and Samy I McFarlane, "Artificial Hallucinations in ChatGPT: Implications in Scientific Writing," *Cureus*, 2023.

Bainbridge, Lisanne, "Ironies of automation," *Automatica*, 1983, *19* (6), 775–779.

Bansal, Gagan, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld, "Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance," in "Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems" 2021, pp. 1–16.

Basiri, Ali, Niosha Behnam, Ruud de Rooij, Lorin Hochstein, Luke Kosewski, Justin Reynolds, and Casey Rosenthal, "Chaos Engineering," *IEEE Software*, 2016, *33* (3), 35–41.

Becker, Joel, Nate Rush, Elizabeth Barnes, and David Rein, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," arXiv preprint arXiv:2507.09089 2025.

Beel, Joeran, Min-Yen Kan, and Moritz Baumgart, "Evaluating Sakana's AI Scientist: Bold Claims, Mixed Results, and a Promising Future?," *ACM SIGIR Forum*, 2025, *59* (1), 1–20.

Bohnet, Bernd, Vinh Q. Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, Tom Kwiatkowski, Ji Ma, Jianmo Ni, Lierni Sestorain Saralegui, Tal Schuster, William W. Cohen, Michael Collins, Dipanjan Das, Donald Metzler, Slav Petrov, and Kellie Webster, "Attributed Question Answering: Evaluation and Modeling for Attributed Large Language Models," arXiv preprint arXiv:2212.08037 2022.

Boiko, Daniil A., Robert MacKnight, Ben Kline, and Gabe Gomes, "Autonomous chemical research with large language models," *Nature*, 2023, *624* (7992), 570–578.

Bran, Andres M., Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller, "Augmenting large language models with chemistry tools," *Nature Machine Intelligence*, 2024, *6* (5), 525–535.

Burckhardt, Sebastian, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, and Christopher S. Meiklejohn, "Durable functions: semantics for stateful serverless," *Proceedings of the ACM on Programming Languages*, 2021, *5* (OOPSLA), 1–27.

Buçinca, Zana, Maja Barbara Malaya, and Krzysztof Z. Gajos, "To Trust or to Think," *Proceedings of the ACM on Human-Computer Interaction*, 2021, *5* (CSCW1), 1–21.

Chambers, Christopher D., "Registered Reports: A new publishing initiative at Cortex," *Cortex*, 2013, *49* (3), 609–610.

_ **and Loukia Tzavella**, "The past, present and future of Registered Reports," *Nature Human Behaviour*, 2021, *6* (1), 29–42.

Chan, Jun Shern, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry, "MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering," in "ICLR 2025" 2024.

Chen, Ziru, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun, "ScienceAgentBench: Toward Rigorous Assessment of Language Agents for Data-Driven Scientific Discovery," in "ICLR 2025" 2024.

Chirigati, Fernando, Rémi Rampin, Dennis Shasha, and Juliana Freire, "ReproZip," in "Proceedings of the 2016 International Conference on Management of Data" 2016, pp. 2085–2088.

Dean, Jeffrey and Luiz André Barroso, "The tail at scale," *Communications of the ACM*, 2013, *56* (2), 74–80.

Ding, Tianyu, Aditya Nannapaneni, Bingfan Liu, and Ling Zhang, "Autonomous Research Agents: A Survey of AI Scientists and the Verification Gap," arXiv preprint arXiv:2608.05179 2026.

Dong, Liming, Qinghua Lu, and Liming Zhu, "AgentOps: Enabling Observability of LLM Agents," arXiv preprint arXiv:2411.05285 2024.

Erol, Mehmet Hamza, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Zou, "Cost-of-Pass: An Economic Framework for Evaluating Language Models," arXiv preprint arXiv:2504.13359 2025.

Gao, Luyu, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu, "RARR: Researching and Revising What Language Models Say, Using Language Models," in "Proceedings

of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)" 2023, pp. 16477–16508.

Gelman, Andrew and Eric Loken, "The Statistical Crisis in Science," *American Scientist*, 2014, *102* (6), 460.

Goddard, Kate, Abdul Roudsari, and Jeremy C Wyatt, "Automation bias: a systematic review of frequency, effect mediators, and mitigators," *Journal of the American Medical Informatics Association*, 2012, *19* (1), 121–127.

Gottweis, Juraj, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan, "Accelerating scientific discovery with Co-Scientist," *Nature*, 2026, *655* (8122), 487–496.

Gray, Jim, "Why Do Computers Stop and What Can Be Done About It?," in "Symposium on Reliability in Distributed Software and Database Systems" 1986.

Gu, Ken, Ruoxi Shang, Ruien Jiang, Keying Kuang, Richard-John Lin, Donghe Lyu, Yue Mao, Youran Pan, Teng Wu, Jiaqian Yu, Yikun Zhang, Tianmai M. Zhang, Lanyi Zhu, Mike A. Merrill, Jeffrey Heer, and Tim Althoff, "BLADE: Benchmarking Language Model Agents for Data-Driven Science," in "EMNLP 2024" 2024.

Helland, Pat, "Idempotence is not a medical condition," *Communications of the ACM*, 2012, *55* (5), 56–65.

Huang, Jie, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou, "Large Language Models Cannot Self-Correct Reasoning Yet," in "ICLR 2024" 2023.

Huang, Qian, Jian Vora, Percy Liang, and Jure Leskovec, "MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation," in "ICML 2024" 2023.

Ifargan, Tal, Lukas Hafner, Maor Kern, Ori Alcalay, and Roy Kishony, "Autonomous LLM-Driven Research — from Data to Human-Verifiable Research Papers," *NEJM AI*, 2025, *2* (1).

Jimenez, Carlos E., John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan, "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?," in "ICLR 2024" 2023.

Jing, Liqiang, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu, "DSBench: How Far Are Data Science Agents from Becoming Data Science Experts?," in "ICLR 2025" 2024.

Kapoor, Sayash, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan, "Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation," arXiv preprint arXiv:2510.11977 2025.

_ , _ , Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan, "AI Agents That Matter," *Transactions on Machine Learning Research*, 2024.

Kerr, Norbert L., "HARKing: Hypothesizing After the Results are Known," *Personality and Social Psychology Review*, 1998, *2* (3), 196–217.

Khanal, Aaditya, Yangyang Tao, and Junxiu Zhou, "Beyond pass@1: A Reliability Science Framework for Long-Horizon LLM Agents," arXiv preprint arXiv:2603.29231 2026.

Krafczyk, M. S., A. Shi, A. Bhaskar, D. Marinov, and V. Stodden, "Learning from reproducing computational results: introducing three principles and the Reproduction Package," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2021, *379* (2197).

Köster, Johannes and Sven Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, 2012, *28* (19), 2520–2522.

Lai, Vivian, Chacha Chen, Alison Smith-Renner, Q. Vera Liao, and Chenhao Tan, "Towards a Science of Human-AI Decision Making: An Overview of Design Space in Empirical Human-Subject Studies," in "2023 ACM Conference on Fairness Accountability and Transparency" 2023, pp. 1369–1385.

Li, Rui, Jia-Chen Gu, Po-Nien Kung, Heming Xia, Junfeng liu, Xiangwen Kong, Zhifang Sui, and Nanyun Peng, "LLM-REVal: Can We Trust LLM Reviewers Yet?," arXiv preprint arXiv:2510.12367 2025.

Liu, Xiao, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang, "AgentBench: Evaluating LLMs as Agents," in "ICLR 2024" 2023.

Lu, Chris, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha, "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery," arXiv preprint arXiv:2408.06292 2024.

_ , _ , _ , Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune, "Towards end-to-end automation of AI research," *Nature*, 2026, *651* (8107), 914–919.

Lubars, Brian and Chenhao Tan, "Ask Not What AI Can Do, But What AI Should Do: Towards a Framework of Task Delegability," in "NeurIPS 2019" 2019.

Luo, Jixiang, "XScientist: A Git-Like Research Protocol for Long-Running Autonomous Scientific Discovery," arXiv preprint arXiv:2607.12301 2026.

Luo, Ziming, Atoosa Kasirzadeh, and Nihar B. Shah, "The More You Automate, the Less You See: Hidden Pitfalls of AI Scientist Systems," arXiv preprint arXiv:2509.08713 2025.

Majumder, Bodhisattwa Prasad, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark, "DiscoveryBench: Towards Data-Driven Discovery with Large Language Models," in "ICLR 2025" 2024.

Min, Sewon, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi, "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation," in "Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing" 2023, pp. 12076–12100.

Missier, Paolo, Khalid Belhajjame, and James Cheney, "The W3C PROV family of specifications for modelling provenance metadata," in "Proceedings of the 16th International Conference on Extending Database Technology" 2013, pp. 773–776.

Mitchener, Ludovico, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C. Landsness, Daniel L. Barabasi, Siddharth Narayanan, Nicky Evans,

Shriya Reddy, Martha Foiani, Aizad Kamal, Leah P. Shriver, Fang Cao, Asmamaw T. Wassie, Jon M. Laurent, Edwin Melville-Green, Mayk Caldas, Albert Bou, Kaleigh F. Roberts, Sladjana Zagorac, Timothy C. Orr, Miranda E. Orr, Kevin J. Zwezdaryk, Ali E. Ghareeb, Laurie McCoy, Bruna Gomes, Euan A. Ashley, Karen E. Duff, Tonio Buonassisi, Tom Rainforth, Randall J. Bateman, Michael Skarlinski, Samuel G. Rodriques, Michaela M. Hinks, and Andrew D. White, "Kosmos: An AI Scientist for Autonomous Discovery," arXiv preprint arXiv:2511.02824 2025.

Munafò, Marcus R., Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis, "A manifesto for reproducible science," *Nature Human Behaviour*, 2017, *1* (1).

National Academies of Sciences, Engineering, and Medicine, *Reproducibility and Replicability in Science*, Washington, DC: The National Academies Press, 2019.

Nosek, B. A., G. Alter, G. C. Banks, D. Borsboom, S. D. Bowman, S. J. Breckler, S. Buck, C. D. Chambers, G. Chin, G. Christensen, M. Contestabile, A. Dafoe, E. Eich, J. Freese, R. Glennerster, D. Goroff, D. P. Green, B. Hesse, M. Humphreys, J. Ishiyama, D. Karlan, A. Kraut, A. Lupia, P. Mabry, T. Madon, N. Malhotra, E. Mayo-Wilson, M. McNutt, E. Miguel, E. Levy Paluck, U. Simonsohn, C. Soderberg, B. A. Spellman, J. Turitto, G. VandenBos, S. Vazire, E. J. Wagenmakers, R. Wilson, and T. Yarkoni, "Promoting an open research culture," *Science*, 2015, *348* (6242), 1422–1425.

Nosek, Brian A., Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor, "The preregistration revolution," *Proceedings of the National Academy of Sciences*, 2018, *115* (11), 2600–2606.

Open Science Collaboration, "Estimating the reproducibility of psychological science," *Science*, 2015, *349* (6251).

Panickssery, Arjun, Samuel R. Bowman, and Shi Feng, "LLM Evaluators Recognize and Favor Their Own Generations," in "NeurIPS 2024" 2024.

Parasuraman, R., T.B. Sheridan, and C.D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 2000, *30* (3), 286–297.

Parasuraman, Raja and Victor Riley, "Humans and Automation: Use, Misuse, Disuse, Abuse," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 1997, *39* (2), 230–253.

Peng, Roger D., "Reproducible Research in Computational Science," *Science*, 2011, *334* (6060), 1226–1227.

Sainz, Oscar, Jon 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" 2023, pp. 10776–10787.

Sakai, Yusuke, Hidetaka Kamigaito, and Taro Watanabe, "HalluCitation Matters: Revealing the Impact of Hallucinated References with 300 Hallucinated Papers in ACL Conferences," in "ACL 2026 (per Semantic Scholar venue record)" 2026.

Sandve, Geir Kjetil, Anton Nekrutenko, James Taylor, and Eivind Hovig, "Ten Simple Rules for Reproducible Computational Research," *PLoS Computational Biology*, 2013, *9* (10), e1003285.

Scheel, Anne M., Mitchell R. M. J. Schijen, and Daniël Lakens, "An Excess of Positive Results: Comparing the Standard Psychology Literature With Registered Reports," *Advances in Methods and Practices in Psychological Science*, 2021, *4* (2).

Schmidgall, Samuel, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum, "Agent Laboratory: Using LLM Agents as Research Assistants," in "Findings of the Association for Computational Linguistics: EMNLP 2025" 2025, pp. 5977–6043.

Siegel, Zachary S., Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan, "CORE-Bench: Fostering the Credibility of Published Research Through a Computational Reproducibility Agent Benchmark," *Transactions on Machine Learning Research*, 2024.

Sigelman, B., L. Barroso, M. Burrows, Patrick Stephenson, M. Plakal, Donald Beaver, Saul Jaspan, and C. Shanbhag, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Technical Report dapper-2010-1, Google 2010.

Simmons, Joseph P., Leif D. Nelson, and Uri Simonsohn, "False-Positive Psychology," *Psychological Science*, 2011, *22* (11), 1359–1366.

Starace, Giulio, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia

Glaese, and Tejal Patwardhan, "PaperBench: Evaluating AI's Ability to Replicate AI Research," in "ICML 2025" 2025.

Stodden, Victoria, Jennifer Seiler, and Zhaokun Ma, "An empirical analysis of journal policy effectiveness for computational reproducibility," *Proceedings of the National Academy of Sciences*, 2018, *115* (11), 2584–2589.

Swanson, Kyle, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou, "The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies," *Nature*, 2025, *646* (8085), 716–723.

Thakur, Aman Singh, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes, "Judging the Judges: Evaluating Alignment and Vulnerabilities in LLMs-as-Judges," in "GEM2 Workshop, ACL 2025" 2024.

Thorne, James, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal, "FEVER: a Large-scale Dataset for Fact Extraction and VERification," in "Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)" 2018, pp. 809–819.

Tommaso, Paolo Di, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, 2017, *35* (4), 316–319.

Trisovic, Ana, Matthew K. Lau, Thomas Pasquier, and Mercè Crosas, "A large-scale study on research code quality and execution," *Scientific Data*, 2022, *9* (1).

Vaccaro, Michelle, Abdullah Almaatouq, and Thomas Malone, "When combinations of humans and AI are useful: A systematic review and meta-analysis," *Nature Human Behaviour*, 2024, *8* (12), 2293–2303.

Wadden, David, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi, "Fact or Fiction: Verifying Scientific Claims," in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)" 2020, pp. 7534–7550.

Walters, William H. and Esther Isabelle Wilder, "Fabrication and errors in the bibliographic citations generated by ChatGPT," *Scientific Reports*, 2023, *13* (1).

Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," in "ICLR 2023" 2022.

Yamada, Yutaro, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha, "The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search," arXiv preprint arXiv:2504.08066 2025.

Yao, Shunyu, Noah Shinn, Pedram Razavi, and Karthik Narasimhan, "

τ

-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains," arXiv preprint arXiv:2406.12045 2024.

Zhao, Zhenyue, Yihe Wang, Toby Stuart, Mathijs De Vaan, Paul Ginsparg, and Yian Yin, "LLM hallucinations in the wild: Large-scale evidence from non-existent citations," arXiv preprint arXiv:2605.07723 2026.

Zheng, Lianmin, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in "NeurIPS 2023 Datasets and Benchmarks" 2023.

Zhu, Yuxuan, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang, "Establishing Best Practices for Building Rigorous Agentic Benchmarks," arXiv preprint arXiv:2507.02825 2025.

Zhuang, Yu, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang, "AgentRewind: Recoverable Execution for Long-Horizon LLM Agents," arXiv preprint arXiv:2608.14380 2026.