AGENT-AUTOBENCH: FROM REAL-WORLD TASKS TO EXECUTABLE BENCHMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Constructing agent benchmarks requires aligning task instructions, obtainable evidence, interaction rules, and scoring criteria while preserving information seeking and clarification. We present Agent-AutoBench, an automated method that transforms heterogeneous real-world materials into versioned executable task packages. Semantic hardening binds reasoning requirements to evidence and rubric criteria, and constrained rewriting updates these dependencies together. Search and fetch tools expose frozen sources, while a disclosure controller governs access to private user facts. Package verification and a tool-using Agent-as-Judge connect construction artifacts with execution feedback for task revision and capability-profile calibration. Daily-Life instantiates the method with a versioned release of 100 selected task packages spanning 11 domains. Archived multi-model evaluations characterize capability coverage and repeated-measurement variability. A historical, non-blind human audit records agreement with 96.47% of 7,815 criterion-level judge decisions. Complementary controlled tests demonstrate criterion-specific responses to output perturbations, restored arithmetic discrimination after a reference correction, and deterministic search and disclosure behavior across two implementations. Together, these results illustrate how evidence control, dependency-aware construction, and execution-grounded diagnosis support executable agent evaluation.

1 INTRODUCTION

A travel-planning agent may receive booking PDFs, weather references, restaurant pointers, and incomplete preferences, yet need to return a consistent itinerary CSV and Markdown report. Constructing an evaluation task from these materials requires deciding which facts the agent can obtain, when they become available, and how they support the score. The generated multi-city travel task in our Daily-Life application illustrates this problem (Section 3; Appendix B).

The construction problem extends beyond generating instructions: reachable evidence, user disclosure, and scoring criteria jointly determine what a task measures. In the travel example, a missed preference could reflect a failure to ask or a fact that the user controller never made obtainable; these require different diagnoses. Supplying every fact removes clarification, while unconstrained simulation can reveal extra facts or help solve the task. Live sources can also change between executions. Lower scores can therefore reflect construction defects as well as solver limitations.

Prior systems establish complementary foundations. AutoBencher optimizes declared benchmark properties through tool-supported question generation (Li et al., 2025). Meta's AutoBench constructs physics problems with programmatic consistency checks (Imani et al., 2025). TaskCraft and TRACE generate or evolve agent tasks using tool context and verification (Shi et al., 2025; Guo et al., 2025). Our setting requires searchable evidence, incomplete user information, and artifact-level scoring to remain consistent, without a formal oracle for every output.

Agent-AutoBench treats the *executable task package* as the construction unit: instructions, obtainable evidence, interaction rules, and scoring targets must remain aligned as the package changes. Three coupled designs implement this idea. **Joint task and rubric construction** shares evidence bindings and updates their dependencies during semantic hardening and constrained rewriting. **Controlled information environments** expose frozen sources through local search and authorize private user facts before a language model renders responses. **Execution-grounded construction feedback**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

combines package checks with artifact- and trace-based judging to guide revision or remeasurement of the same version.

The contribution is a construction method and system for tasks whose reasoning, information access, and scoring depend on one another. The package representation makes those dependencies explicit; local retrieval and staged disclosure make information seeking part of execution; and evidence-seeking judging connects observable behavior to construction feedback. Daily-Life instantiates the method through a versioned task release; separately archived multi-model evaluations characterize its use in the application. Section 3 organizes the evidence around task release and coverage, scoring reliability, and observed model performance and repeats. Historical non-blind review and completed criterion-specific controls provide complementary evidence about the scoring process. The appendices document the implementation routes, task artifacts, historical cohorts, and complete diagnostic-control accounting.

2 Joint Construction of Executable Task Environments

2.1 Source-grounded task and rubric construction

Given source materials and target capabilities, Agent-AutoBench constructs a versioned bundle $b_v = (u_v, m_v, e_v, o_v, r_v)$: instructions, accessible materials, an execution environment, private reference targets, and a weighted rubric. This conceptual grouping includes tool interfaces and, for interactive tasks, controller-side user state in e_v. The solver accesses permitted files, retrieval results, and authorized user responses; reference answers, rubrics, and undisclosed user facts remain evaluation-private. The author changes the bundle; the solver executes through public interfaces; the judge scores outputs using private criteria; the construction controller decides revision or remeasurement. A seed fixes source pointers, domain, capabilities, and authoring mode (Appendix A); grounding records provenance and content hashes.

Joint construction means coordinated authoring under shared evidence bindings. The task author creates instructions, constraints, and material specifications; the rubric author uses that context to derive weighted criteria, check points, failure examples, and reference targets. Supplied rubrics constrain conversion; regeneration can rederive factual values. Bindings connect requirements and criteria to accessible evidence or declared derivations, so revisions can update their dependencies together. Figure 1 summarizes the operations; Appendix A specifies their execution routes.

2.2 Semantic hardening and constrained rewriting

Semantic hardening targets reasoning structure: resolving source conflicts, intersecting constraints, combining evidence, applying a reference date, disambiguating entities, or recognizing an unsupported subgoal. Authoring must retain an actionable core: a missing subgoal can require justified abstention while leaving a feasible response. The author declares each mechanism for rubric generation; audit fields bind it to supporting evidence, distractors, or an expected resolution. These records make the intended reasoning challenge inspectable alongside its evidence.

Rewriting operates on the whole bundle. Regeneration may reorganize a request; conversion preserves supplied scoring intent without automatically injecting difficulty. Feedback-directed hardening retains the scenario and applicable constraints while changing the challenge. Corrective rewriting targets diagnosed inconsistencies and carries forward compatible requirements. Both regenerate dependent criteria and reference bindings. Appendix A.5 specifies the preservation instructions and checks. Refreshing a source capture changes the evidence version; it is not remeasurement of an unchanged task.

2.3 Local search MCP: controlled evidence access

The environment separates supplied files from a searchable frozen web corpus. A local Model Context Protocol (MCP) server exposes `search(query)`, returning titles, URLs, and snippets, and `fetch(url)`, returning captured page text. Stored indices and normalized URLs resolve requests. For retrieval-based tasks, agents must search, select, and read evidence rather than receive every

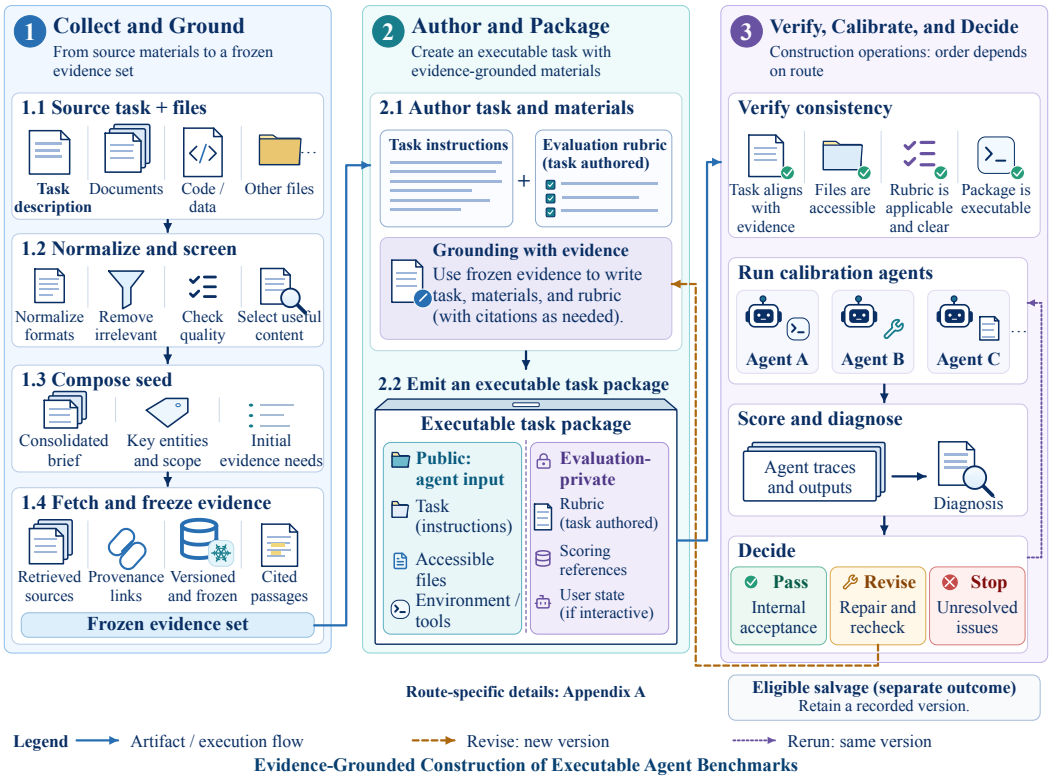

Figure 1: **Agent-AutoBench construction workflow.** Shared grounding links task and rubric authoring (middle); the emitted package separates public tools, including local search, from private rubric and user state (middle bottom); execution and diagnosis guide construction feedback (right). For interactive tasks, a disclosure controller mediates access to private user state. Revision changes the package; remeasurement keeps the same version. Appendix A specifies route-dependent check order and distinguishes internal acceptance and salvage from independent usability assessment.

reference in the initial prompt. The solver's task-fact policy is

$$E_{\text{local}} \succ E_{\text{live}} \succ K_{\text{model}}, \tag{1}$$

where $\succ$ denotes precedence, not measured source quality. Local materials fix task-specific facts; enabled live search supplies missing background; model memory cannot overwrite frozen values or replace required retrieval. Online search is advertised only when attached. A missing index or captured file is an environment error, distinct from a query with no match. Freezing controls available content, while retrieval behavior, solver sampling, judging, and live fallback can still change scores.

2.4 Controlled user simulation with staged information

Interactive tasks carry private facts $\mathcal{H}$ and disclosure rules. With visible history $h_{\le t}$ and controller state σ_t (rules, prior disclosures, and triggers), information release and response rendering are separate operations:

$$\mathcal{D}_t = \text{Disclose}(\sigma_t, h_{\le t}) \subseteq \mathcal{H}, \qquad y_t = \text{Render}(h_{\le t}, \mathcal{D}_t). \tag{2}$$

Here $\mathcal{D}_t$ includes facts authorized for this response, including prior disclosures; y_t is the response. Fixed task configuration and persona context are implicit. The rule-based controller handles opening, question/topic, and progress triggers; never-disclosed facts remain unavailable. Render combines language generation, response checks, bounded rewriting, and template fallback. Its language model receives authorized content rather than selecting facts from the private specification; response checks retain controller-side access to private facts. Equation 2 separates the disclosure decision from language variation; it is not a guarantee that all generated text obeys that decision. Checks target unauthorized matches and missing numeric or filename anchors. Logged triggers distinguish elicited facts from volunteered ones.

2.5 VERIFICATION AND EVIDENCE-SEEKING AGENT-AS-JUDGE

The verifier inspects the package: required structure, reachable materials, reference bindings, supported leakage patterns, and evidence for declared difficulty mechanisms. For a declared source conflict, it checks support for the stated values; this necessary condition does not itself prove incompatibility or reasoning difficulty. Results retain failures, warnings, and unsupported checks, separating evidence for individual checks from aggregate acceptance. Verification follows batch emission; the calibration route also checks grounding before retention.

Agent-as-Judge evaluates a solver run using private criteria and reference targets, inspecting delivered artifacts, tool results, traces, and dialogue. File presence alone cannot establish content correctness or visual quality. The judge returns criterion verdicts, reasons, and evidence anchors. *Construction diagnosis* interprets this evidence as feedback about the package, solver behavior, or an unresolved cause; a low score need not imply a task defect or require revision.

The batch route shares inspection across criteria and aggregates repeated ballots. For a fixed task and evaluated run with a binary rubric, let $\widehat{y}_{a,c} \in \{0, 1, \perp\}$ denote the aggregated fail, pass, or explicit exclusion for solver a and criterion c, with authored weight $w_c > 0$. When $\sum_{c \in \mathcal{K}_a} w_c > 0$, the reported score is

$$q_a = \mathrm{round}_3\left(\frac{\sum_{c \in \mathcal{K}_a} w_c \widehat{y}_{a,c}}{\sum_{c \in \mathcal{K}_a} w_c}\right), \qquad \mathcal{K}_a = \{c : \widehat{y}_{a,c} \neq \perp\}. \tag{3}$$

Exclusion changes the denominator: scores based on different graded criterion sets need not be comparable. A score alone cannot establish rubric coverage. Equation 3 summarizes graded outcomes, not missing executions. The evidence-sensitive voting rule and ordinal grading are specified in Appendix A.5; the aggregation policy allows evidence-grounded dissent to override a shallow pass majority. Rubric coverage, verdict completeness, and availability of every calibration score remain distinct. Agentic grading builds on prior work (Zhuge et al., 2025); here judgments guide construction, while criterion-specific controls probe scorer behavior.

2.6 CAPABILITY-PROFILE CALIBRATION AND FEEDBACK

For normalized solver score $q_a \in [0, 1]$ and configured tiers $\mathcal{A}_1, \ldots, \mathcal{A}_T$, indexed from intended weaker to stronger capability, the *construction controller* uses $z_j = \max_{a \in \mathcal{A}_j} q_a$, the best score in tier j. With profile filtering enabled and all required solver scores available, it requires

$$\ell \leq z_T \leq \tau, \qquad z_{j+1} \geq z_j \ (j < T), \qquad z_T - z_1 \geq \delta. \tag{4}$$

Here ℓ and τ bound the strongest tier's target score; δ sets minimum endpoint separation. Adjacent tiers may tie. This agent-tier score profile is a construction target, distinct from the capability-subset matrix reported later.

Posterior hardening and candidate selection. Feedback identifies criteria the strongest solver still passes as tightening targets; failures caused by inaccessible materials instead request repair. The revision prompt carries the prior task, check points, failure examples, expected metrics, and tolerances. Repair takes precedence over hardening while retaining compatible prior requirements. The Daily-Life construction route additionally continues eligible single-turn tasks beyond the first profile pass, within the round budget. Among qualified snapshots it selects the lowest strongest-tier score, breaking ties by larger tier separation, and restores the selected specification and emitted package together. Criterion-retention diagnostics flag lost requirements without enforcing semantic monotonicity; selecting a low measured score does not itself establish valid difficulty.

Diagnosis guides changes to materials, requirements, or difficulty. Unstable calibration can instead trigger fresh executions of the same version, using per-agent median scores. Grounding and computed-reference checks can request corrections before retention. Internal acceptance, eligible budget salvage, and termination are separate outcomes. Some correction paths retain decisions based on earlier scores; final-version calibration therefore requires fresh execution. Independent usability review is separate from these internal decisions. Appendix A details version changes, calibration branches, and the evidence available at each decision.

3 EVALUATION: TASK COVERAGE, SCORING, AND MODEL BEHAVIOR

Our evaluation separates three questions: what the system constructs, how its judge responds to controlled violations, and what behavior the generated tasks expose. The release inventory measures package coverage; historical review measures endorsement; fresh controls test criterion sensitivity; archived solver scores describe model behavior. Table 1 summarizes the question answered by each evidence source; Appendix D gives cohort membership and version linkage. Construction mechanisms and measured endpoints are reported separately.

Table 1: **Evidence for distinct evaluation questions.** Each row uses its own observation unit; the studies are complementary rather than a pooled experiment.

Evidence	Question	Measured endpoint
Versioned task release	What does the system construct?	Package inventory, domain and mode coverage.
Retrieval/disclosure checks	Do information-access rules execute as specified?	Fixture-level source-function observations.
Scorer and reference controls	Does changing the relevant evidence change the targeted judgment?	Fixed-output criterion verdicts.
Historical human review	Do reviewers endorse the recorded judgments?	Non-blind criterion-level agreement.
Archived solver evaluations	What model behavior do the tasks expose?	Capability-subset scores and repeat spans.

3.1 CONSTRUCTED TASK RELEASE AND COVERAGE

Daily-Life instantiates Agent-AutoBench with heterogeneous everyday requests. One archived, versioned release contains 100 selected packages across 11 domains: 39 scratch-generated, 37 regenerated, and 24 converted from supplied tasks. Each package contains metadata, a seed, instructions, a rubric, reference files, and an identifier-matched authoring specification. Human contributions enter source materials; model scores and domain quotas determine release membership. The inventory records selected packages and preserves anonymous identifiers and artifact hashes (Appendix B).

An executable travel-planning environment. The generated travel task in Table 2 follows one evidence constraint through the package. Its saved specification records restaurant references without verifiable merchant details. The task requires those gaps to remain explicit in the itinerary CSV and report; linked rubric criteria check that unsupported facts are not invented and unresolved information is not presented as confirmed. This example makes the construction mechanism inspectable: one evidence gap becomes an output obligation and three linked checks, rather than an unsupported reference answer. The table paraphrases this saved version, whose artifact scope is specified in Appendix B.

Table 2: **One source-bound requirement in a generated travel task.** English paraphrases of the same saved specification connect available evidence, observable outputs, and rubric criteria.

Stage	Task artifact and required behavior
Evidence and constraint	Restaurant-search bindings record missing merchant details. Names, ratings, and per-person prices must remain unresolved rather than be supplied from model memory.
Observable output	Pending items receive separate itinerary-CSV records with an explicit status. The Markdown report lists dining-information gaps and fields needing verification.
Linked checks	Criterion Y4 checks abstention from invented restaurant facts; E7 and C5 check that the uncertainty remains explicit in the CSV and report, respectively.

Executable information-access checks. Frozen synthetic queries and dialogue sequences exercise the production retrieval and disclosure controllers. Exact retrieval, a no-match query, plain-text fetch,

authorized release, and withholding tests pass in both implementation snapshots. The anonymous supplement replays the actual source functions offline; Appendix C.3 gives every check and distinguishes controller decisions from generated replies.

3.2 SCORING RELIABILITY EVIDENCE

Historical criterion review. Reviewers saw machine decisions and recorded agreement or disagreement, providing a non-blind audit of criterion-level judgments. In a platform snapshot retrieved September 20, 2026, 9 reviewers endorsed 7,539 of 7,815 criterion-level decisions across 216 distinct item identifiers and 300 platform tasks: **96.47% reviewer endorsement**, with an item-cluster bootstrap interval of $[95.88, 97.03]\%$ at the 95% level. The cohort includes all named Daily-Life project versions; its latest submitted review dates to August 27, 2026. Restricting to explicit pass/fail judgments gives 96.44%; platform-QA-approved records give 96.20%. Appendix D.4 audits the denominator and preserves the earlier snapshot.

Criterion-specific scorer controls. A normal PDF, a version with masked table cells, and a version with section markers removed test table and layout criteria in a subsequent scorer version. Visually masking table cells changes the table criterion from pass to fail while preserving the layout verdict; removing section markers changes the layout verdict while preserving the table verdict. The masked PDF retains extractable text, making the contrast informative about visual evidence use. Inputs, rubrics, and expected contrasts were frozen before scoring, and expected outcomes were withheld from the judge. HTML controls also exercise interactive behavior; their failed on-demand browsing and incomplete ballots are retained in Appendix C alongside every PDF outcome. The historical review measures endorsement of visible decisions, while these controls probe responses to specific violations under a later scorer configuration.

Controlled reference repair. A selected travel task provides a direct intervention on package consistency. Its declared visit durations sum to 380 minutes, but the rubric and private reference specify 320. We corrected only those linked targets and rescored the same incorrect and correct outputs under a fixed native judge configuration: 4 fresh conditions, with 1 ballot each. Table 3 shows that the original target reverses the arithmetic verdicts, while the corrected target restores the intended discrimination. This purposively selected intervention isolates reference consistency; it is not an estimate of autonomous construction benefit. Appendix C.2 documents the frozen protocol and evidence.

Table 3: **Reference consistency changes criterion discrimination.** Native arithmetic verdicts for fixed outputs in the same selected task.

Scoring target	Incorrect output (320 min)	Correct output (380 min)
Original	Pass	Fail
Evidence-corrected	Fail	Pass

3.3 OBSERVED MODEL PERFORMANCE AND REPEATS

The capability panel records model performance across task subsets. Public model names and recorded configuration labels are listed in Appendix D.6. Its published aggregate scores summarize subset performance, while raw common-item comparisons characterize observed pairwise separation. These complementary views expose both coverage and overlap in the archived measurements. Figure 3 shows both panels. Appendix D specifies score adjustments, overlap, sampling, and the additional statistical panels.

Observed scores vary across the signature-matched groups. The selected panel contains 305 criterion-signature-matched item–solver–round groups and 876 scores, with median observed range 0.148. In 73 groups (23.9%), scores straddle $\tau = 0.55$ (Figure 2). These observed spans motivate retaining repeated measurements and distinguishing remeasurement from package revision. Appendix D details grouping and the separate recovered repeat panel.

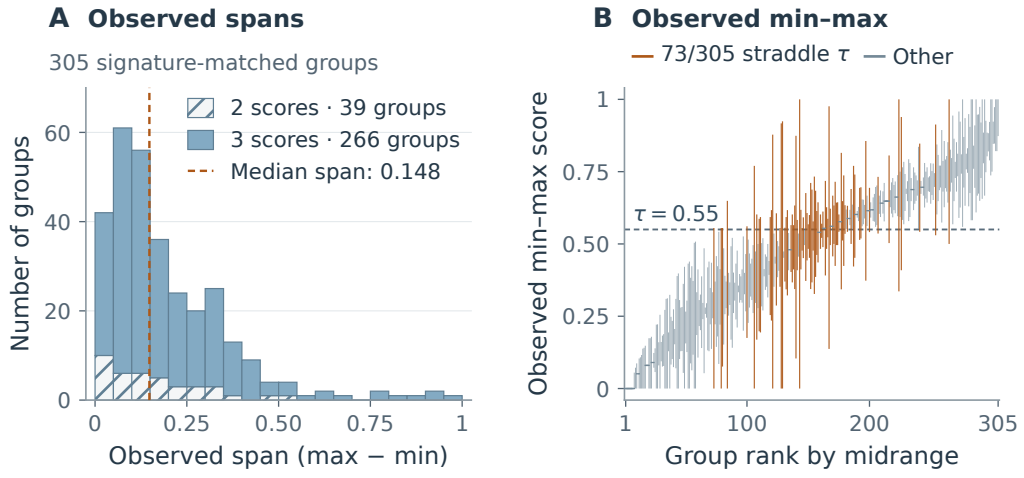

Figure 2: **Observed spans in selected archival repeat groups.** Panel A bins observed spans, stacking counts for groups with two and three scores. Panel B sorts the observed min–max spans by midrange. The 305 signature-matched item–solver–round groups contain 876 scores; 73 groups satisfy $\min(s) \leq 0.55 < \max(s)$. Segments show observed minima and maxima, not confidence intervals. Groups were selected through adaptive repetition, can share an item, and match criterion IDs and weights without proving identical tasks or rubric text. Scalar crossing does not establish reversal of the full multi-condition gate.

4 Related Work

Benchmark construction and feedback. AutoBencher (Li et al., 2025) optimizes dataset descriptions for declared benchmark properties. BenchAgents (Butt et al., 2024) coordinates planning, generation, verification, and evaluation with developer feedback. Autodata (Kulikov et al., 2026) uses a challenger, weak and strong solvers, and a verifier or judge to revise data-generation recipes. Agent-AutoBench couples construction feedback to shared evidence bindings and controlled information access in executable task packages.

Programmatic and formal consistency. Meta's AutoBench (Imani et al., 2025) generates physics benchmarks using parameterized Python solutions, dimensional and numerical checks, and corrective feedback. Anchor (Ivanov & Rana, 2026) derives instructions, environments, solutions, and verifiers from a shared constraint-optimization specification. Agent-AutoBench studies heterogeneous task materials without assuming a formal oracle for every artifact; task usability and scorer correctness remain distinct evaluation targets.

Agent-task synthesis and evolution. TaskCraft (Shi et al., 2025) expands tool-grounded atomic tasks in depth and width with incremental verification. AgentSynth (Xie et al., 2025) composes executed subtasks into computer-use tasks. AutoPlay (Ramrakhya et al., 2025) uses environment exploration to generate tasks and tests their utility for agent training. TRACE (Guo et al., 2025) evolves existing benchmark tasks with recorded trajectories and reproduction-based validation. Self-Challenging Language Model Agents (Zhou et al., 2025) couples generated instructions with verification functions and successful and failing examples.

Reliable agent evaluation. AndroidWorld (Rawles et al., 2025) distinguishes repeated execution of identical tasks from variation in task parameters. τ-bench (Yao et al., 2025) evaluates consistent success across repeated user–agent interactions. HAL (Kapoor et al., 2025) jointly analyzes models, scaffolds, costs, and execution logs. Agent-as-a-Judge (Zhuge et al., 2025) uses agentic evaluation to inspect agent behavior and progress. Our tool-using judge operates with authored rubrics and task environments, linking criterion-level evidence to package diagnosis and revision.

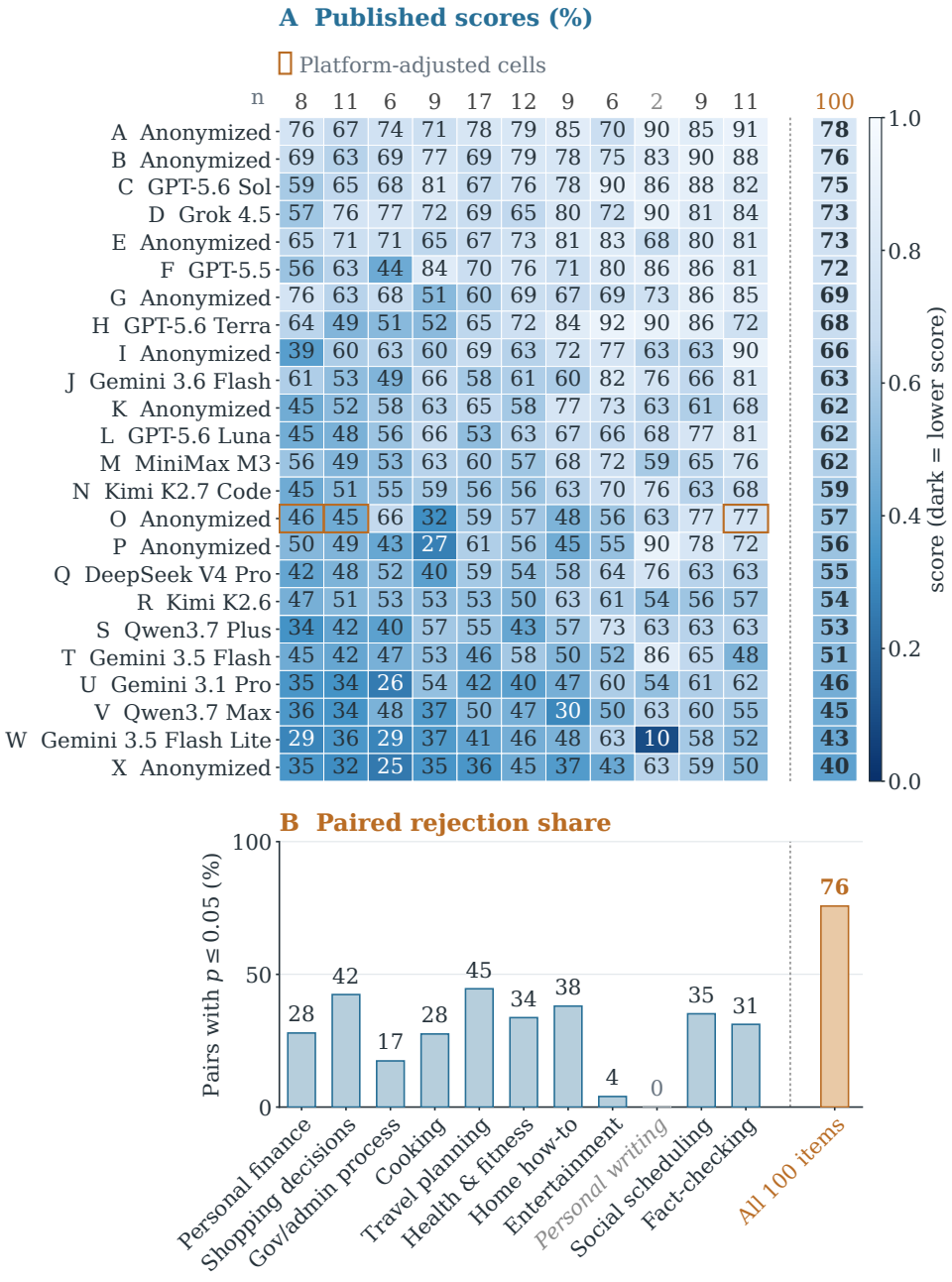

Figure 3: **Archival capability coverage.** (A) Published scores for 24 recorded solver configurations across 11 subsets and the 100-item aggregate. Orange outlines mark platform adjustments; header n gives declared subset size. (B) Percentage of 276 solver pairs with nominal paired sign-flip $p \leq 0.05$ on raw common-item scores, excluding sentinel rows. Hatching denotes subsets too small to attain this threshold. These descriptive shares have no multiplicity correction and do not establish a model ordering. Rows show recorded public model names where available and fixed IDs otherwise; Table 12 gives configuration suffixes, references, and exact aggregate scores. Appendix D gives overlap and sampling details.

5 LIMITATIONS

The release, capability panel, adaptive repeats, and historical review are separate cohorts without an established common release-version join. Release counts measure selected packages. Non-blind endorsement measures agreement with visible historical decisions; the later scorer controls measure sensitivity to specific violations. Neither estimates current scoring accuracy against independent reference judgments. Repeat signatures do not authenticate full task versions, and performance panels remain descriptive. Frozen evidence fixes available content without establishing its truth; live fallback, solver sampling, and judging introduce variability. Disclosure checks leave semantic leakage and conversational realism unresolved. Calibration depends on configured solver tiers. General scoring accuracy, component effects, and performance on other models and task populations require independent reference judgments and broader evaluation.

6 CONCLUSION

Agent-AutoBench turns heterogeneous task materials into versioned executable benchmarks by jointly constructing instructions, evidence access, interaction rules, and private scoring criteria. Shared evidence bindings support semantic hardening and coordinated revision; local retrieval and staged disclosure preserve information seeking; artifact- and trace-based judging connects execution to construction feedback. Daily-Life instantiates this design in an archived task release with multi-model measurements. Historical criterion review and completed scorer controls provide complementary evidence about judging, while capability profiles and repeated scores characterize observed agent behavior. Together, the system and its documented artifacts make the dependencies between task construction and agent evaluation explicit and inspectable.

AI USE STATEMENT

AI tools assisted manuscript review, literature discovery, implementation, and feedback on study design and analysis. The benchmark pipeline uses language models for task generation and user simulation, as described in the methodology. The authors take responsibility for the manuscript, analyses, and generated artifacts.

REPRODUCIBILITY STATEMENT

An anonymous supplementary archive provides the capability matrix, anonymized per-item scores, plotting and verification scripts, the paired reference-repair diagnostic, and an offline replay of the production retrieval and disclosure checks. Reported quantities derive from a frozen claim ledger with source bindings and generated numerical macros. The appendices document task artifacts, cohort definitions, scoring rules, analysis assumptions, and diagnostic outcomes. Archived inventories and content hashes identify the underlying versions; access to source materials and execution records is subject to their availability and redistribution rights. The historical analyses retain missing observations and distinguish available artifact bindings from unverified full-version joins.

ETHICS STATEMENT

Source materials require appropriate access and redistribution rights. Public releases must remove personal data and internal identifiers while preserving task semantics. Evaluation environments should expose only the tools and data required by the task. Human review protocols should document consent, compensation, and adjudication procedures.

REFERENCES

Alibaba Cloud. Qwen3.7-Plus: Multimodal Agent Intelligence, 2026. URL https://www.alibabacloud.com/blog/qwen3-7-plus-multimodal-agent-intelligence_603206. Official model documentation; accessed September 26, 2026.

Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. BenchAgents: Multi-agent systems for structured benchmark creation, 2024. URL https://arxiv.org/abs/2410.22584v2.

DeepSeek-AI. DeepSeek-V4-Pro: Model Card, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Official model documentation; accessed September 26, 2026.

Google. Gemini 3.6 Flash, 2026a. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.6-flash. Official model documentation; accessed September 26, 2026.

Google. Gemini 3.5 Flash, 2026b. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash. Official model documentation; accessed September 26, 2026.

Google. Gemini 3.5 Flash-Lite, 2026c. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash-lite. Official model documentation; accessed September 26, 2026.

Google DeepMind. Gemini 3.1 Pro: Model Card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/. Official model documentation; accessed September 26, 2026.

Dadi Guo, Tianyi Zhou, Dongrui Liu, Chen Qian, Qihan Ren, Shuai Shao, Zhiyuan Fan, Yi R. Fung, Kun Wang, Linfeng Zhang, and Jing Shao. Towards self-evolving benchmarks: Synthesizing agent trajectories via test-time exploration under validate-by-reproduce paradigm. *arXiv preprint arXiv:2510.00415v1*, 2025. URL https://arxiv.org/abs/2510.00415v1. Version consulted; accepted at ICLR 2026 as Towards Self-Evolving Agent Benchmarks: Validatable Agent Trajectory via Test-Time Exploration.

Shima Imani, Seungwhan Moon, Adel Ahmadyan, Lu Zhang, Ahmed Kirmani, and Babak Damavandi. AutoBench: A dynamic benchmark for auditing scientific reasoning in large language models. EMACS at CVPR 2025, 2025. URL https://openreview.net/pdf?id=HJ0U8GaP0m.

Maksim Ivanov and Abhijay Rana. Anchor: Mitigating artifact drift in agent benchmark generation, 2026. URL https://arxiv.org/abs/2605.26321v1.

Sayash Kapoor, 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:2510.11977v1*, 2025. URL https://arxiv.org/abs/2510.11977v1. Version consulted; accepted at ICLR 2026.

Ilia Kulikov, Chenxi Whitehouse, Tianhao Wu, Yixin Nie, Swarnadeep Saha, Eryk Helenowski, Weizhe Yuan, Olga Golovneva, Jack Lanchantin, Yoram Bachrach, Jakob Foerster, Xian Li, Han Fang, Sainbayar Sukhbaatar, and Jason Weston. Autodata: An agentic data scientist to create high quality synthetic data, 2026. URL https://arxiv.org/abs/2606.25996v3.

Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. Autobencher: Towards declarative benchmark construction. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/eb216114f3eaad22506fd1bc7bbff0ca-Abstract-Conference.html.

MiniMax. MiniMax M3: Frontier Coding, 1M Context, Native Multimodality—All in One Model, 2026. URL https://www.minimax.io/blog/minimax-m3. Official model documentation; accessed September 26, 2026.

Moonshot AI. Kimi-K2.7-Code: Model Card, 2026a. URL https://huggingface.co/moonshotai/Kimi-K2.7-Code. Official model documentation; accessed September 26, 2026.

Moonshot AI. Kimi-K2.6: Model Card, 2026b. URL https://huggingface.co/moonshotai/Kimi-K2.6. Official model documentation; accessed September 26, 2026.

OpenAI. GPT-5.6 Sol, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.6-sol. Official model documentation; accessed September 26, 2026.

OpenAI. Introducing GPT-5.5, 2026b. URL https://openai.com/index/introducing-gpt-5-5/. Official model documentation; accessed September 26, 2026.

OpenAI. GPT-5.6 Terra, 2026c. URL https://developers.openai.com/api/docs/models/gpt-5.6-terra. Official model documentation; accessed September 26, 2026.

OpenAI. GPT-5.6 Luna, 2026d. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Official model documentation; accessed September 26, 2026.

Qwen Team. Qwen3.7: The Agent Frontier, 2026. URL https://qwen.ai/blog?id=qwen3.7&locale=en. Official model documentation; accessed September 26, 2026.

Ram Ramrakhya, Andrew Szot, Omar Attia, Yuhao Yang, Anh Nguyen, Bogdan Mazoure, Zhe Gan, Harsh Agrawal, and Alexander Toshev. Scaling synthetic task generation for agents via exploration. *arXiv preprint arXiv:2509.25047v1*, 2025. URL https://arxiv.org/abs/2509.25047v1. Version consulted; accepted at ICLR 2026.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks, 2025. URL https://arxiv.org/abs/2506.10055v2.

SpaceXAI. Introducing Grok 4.5, 2026. URL https://x.ai/news/grok-4-5. Official model documentation; accessed September 26, 2026.

Jingxu Xie, Dylan Xu, Xuandong Zhao, and Dawn Song. AgentSynth: Scalable task generation for generalist computer-use agents, 2025. URL https://arxiv.org/abs/2506.14205v2.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-Challenging Language Model Agents, 2025. URL https://arxiv.org/abs/2506.01716v1.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.

APPENDIX GUIDE

Appendix A specifies the construction operations, evidence-access rules, scoring, and version semantics, followed by commands for reproducing the public analyses. Appendix B presents the release inventory and task examples. Appendix C documents controlled scorer interventions and retrieval/disclosure checks. Appendix D provides cohort definitions, historical analyses, and model-configuration provenance.

A IMPLEMENTATION DETAILS AND VERSION SEMANTICS

Figure 4 complements the main workflow by showing how evidence, disclosure, and judgment connect within the task package.

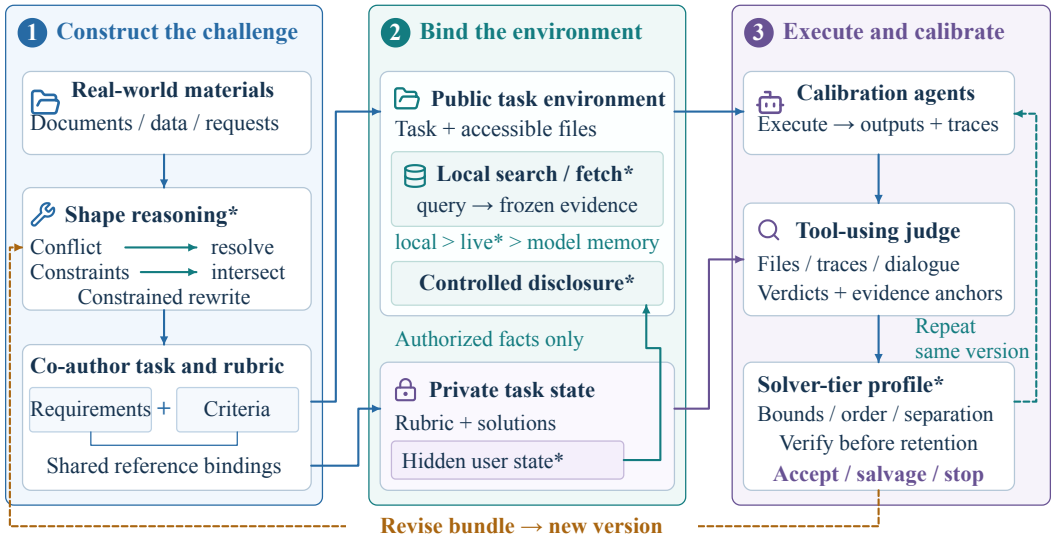

Figure 4: **Complementary view of the joint construction design.** Shared reference bindings connect task requirements, reasoning challenges, and rubric criteria. Public inputs include frozen retrieval and controlled disclosure for applicable tasks; private state supplies authorized user facts and judge criteria. Agents produce artifacts and traces for tool-based judging. Internal calibration either retains an output, revises the bundle into a new version, or repeats execution of the same version. The evidence-priority relation governs task facts, not measured model quality. Configuration-dependent paths are marked with an asterisk. Internal retention precedes independent usability assessment.

The following details expand the configurable operations in Section 2. They describe system routes and the version boundaries of their artifacts. Optional source expansion creates additional seeds and is accounted for separately from the default construction route.

A.1 EVIDENCE CAPTURE AND TASK AUTHORING

Captured evidence and version identity. Grounding resolves source pointers into local evidence snapshots. Captured documents and attachments carry their source location, local file location, content hash, and retrieval metadata. These records fix the evidence used during authoring and allow later checks against the same captured content. Refreshing a source during repair updates the capture and its retrieval record, creating a changed evidence version. Frozen therefore refers to a captured version, not an unchangeable construction history. Self-contained tasks can proceed without external retrieval; live tool access, when enabled, remains part of the execution configuration. Source reliability and the correctness of derived reference values are assessed separately from snapshot identity.

Package fields and private state. The bundle in Section 2 is a conceptual grouping, not the storage schema: `simulator_spec` is a separate specification field, emitted under `evaluation/simulator/spec.json` and read by the disclosure controller. Its private facts are not judge-only state.

Task and rubric responsibilities. The task author uses the seed and frozen evidence to produce the instruction, constraints, materials, and executable specification. Within this step, a dedicated rubric-generation component receives the task context, reference bindings, and available evidence. It produces weighted criteria with check points, failure examples, and derived reference targets, and can revise them in response to bounded quality feedback. Supplied author rubrics have a mode-specific role: conversion treats them as an alignment constraint, whereas regeneration uses them as a reference and rederives factual values from the evidence. These are distinct authoring responsibilities; agreement between generators is recorded separately from independent validation.

A.2 Verification, calibration, and version changes

Build verification and repair. The default build route is composition, grounding, task and rubric authoring, emission, and verification. The verifier returns `PASS`, `WARN`, or `FAIL`, together with reasons and a failure category. Its checks cover required structure, reference bindings, evidence accessibility, and supported forms of answer leakage. For example, missing required input material and a supplied file matching a reference answer require different repairs. A material format that the leakage scanner cannot read is reported as a warning. The default batch route permits warnings and attempts category-specific repair for selected failures before rerunning the build and verifier; remaining failures stop that source attempt. Holistic quality review is an additional, explicitly invoked operation.

Configured tier profile. The tier ordering and thresholds are construction configuration choices, not a model ranking inferred from a generated task. Ordinary calibration uses the nondecreasing full-vector condition in Equation 4; eligible posterior salvage additionally checks strict ordering between the weakest and strongest retained scores. Neither establishes independent validity.

Resampling the same task version. The construction controller can collect additional executions when tiers invert, the strongest-tier score lies just above the ceiling, or a judge-disagreement diagnostic overlaps it. This branch preserves the task version, takes the median of valid scores for each agent, and reevaluates the tier profile. If any agent has fewer than two valid samples, it retains the preceding decision. The disagreement diagnostic is derived from split criterion votes and serves as a resampling trigger, not a confidence interval.

Revising task artifacts. Revision instead changes the candidate artifacts. Feedback combines the preceding instruction and rubric with criterion verdicts, judge reasons, and trace evidence such as failed material reads. A failure accompanied by an inaccessible attachment directs repair toward material access before further difficulty adjustment. Other feedback can revise task difficulty or the targeted criteria. The author then regenerates the specification and the emitter writes the revised package. Table 4 summarizes these distinct operations.

Table 4: **Implemented diagnostic operations.** Observed conditions trigger artifact repair, repeated execution, revision, retention checks, or stopping.

Observed condition	Operation and affected object
Material or verification defect	Repair relevant artifacts; rebuild and reverify.
Unstable calibration signal	Collect new executions of the same task version.
Difficulty profile missed	Revise task artifacts using criterion and trace feedback.
Calibration condition met	Check grounding and computed references before retention.
Limit or unresolved failure	Stop, or salvage an eligible recorded version.

Reference checks, rescoring, and version records. Before acceptance or salvage, grounding checks and a separate model-based recomputation of declared computed targets can trigger further

repair. Confirmed numerical mismatches are distinguished from values the recomputation cannot establish. Certain rubric repairs also request rescoring of existing trajectories; rescoring is distinct from a new agent execution. In this path, the calibration decision can still use pre-repair scores; fresh execution is needed to attribute calibration to the final version. Round snapshots preserve specifications and calibration summaries, and execution records contain an execution identifier and hashes of the instruction, rubric, materials, search corpus, simulator specification, and oracle manifest.

Stopping and retention outcomes. The loop is bounded by its round limit and, when enabled, a wall-clock limit. It can accept a calibrated candidate, salvage an eligible recorded version, or terminate without retaining one. Salvage remains a separate internal outcome. Version-linked records distinguish the retained artifacts from the versions that supplied calibration scores; unresolved joins remain explicit.

A.3 Local evidence access and its limits

The emitter builds a search index of candidate titles, URLs, and snippets and a URL-to-page index over successful nonempty captures. The MCP search server normalizes case and whitespace, attempts an exact key match, then uses token and Chinese-substring scores with a unique-best-match condition in the main implementation. The Daily-Life implementation instead merges tied best matches, deduplicating by URL. Fetch normalizes the requested URL and reads the stored page body. For fixed indices and query, these operations follow fixed code paths; they do not invoke a language model to generate retrieval results. An unmatched query, an absent index, and a missing indexed page have different statuses.

The corpus is placed outside the ordinary workspace-file presentation to elicit search/fetch use. This placement is an interaction convention, not a security guarantee against arbitrary filesystem exploration. Its contents are permitted task evidence. Rubrics, answers, and simulator-private facts have a separate exclusion boundary. Online search is attached at runtime when available, with the solver prompt generated from the same availability decision. Without that tool the prompt reports the missing channel instead of asking the model to imitate web retrieval. Live results must not overwrite frozen task-specific values. The precedence rule is in the solver prompt; the judge receives the task's rubric and reference evidence, rather than an assumed identical prompt rule.

This policy defines the benchmark's information setting. It does not claim that a local page is universally more accurate than a current online source or that agents always obey the instruction. Captured content can be wrong or outdated. Retrieval replayability, evidence-policy adherence, and score stability are therefore separate measurements.

A.4 Disclosure control and conversational rendering

Hidden facts support opening disclosure, keyword-triggered questions or topics, and never-disclosed fields. Progress rules can volunteer eligible facts after delayed clarification or a stall, with their reasons retained in the trace. Given the same specification and message history, the disclosure controller is rule based; different agent utterances can lead to different information states. Response generation remains model based. The implementation tests unauthorized lexical, entity, and number matches and checks that required numeric or filename anchors are actually conveyed. Failed checks can trigger rewriting or a template carrying the authorized values. Suspected simulator assistance is logged for review. These checks do not prove complete semantic coverage or human-equivalent behavior.

For supported DOCX and XLSX outputs, an optional user-feedback step presents image previews reconstructed from extracted text and cell values to a model, together with the visible dialogue. These bounded previews support content-level feedback but do not preserve native document layout or establish formatting quality; Agent-as-Judge requires separate rendered evidence for layout assessment.

A.5 Declared reasoning mechanisms and verification limits

Declared mechanisms and their evidence checks. The author declares planted mechanisms in the specification; the rubric author receives these declarations alongside source captures, visible material

Table 5: **Semantic difficulty mechanisms and bounded checks.** Corpus checks inspect concrete support for declared mechanisms; they do not prove reasoning depth, uniqueness, or exhaustive absence of an answer.

Mechanism	Intended challenge	Implemented evidence check
Source conflict	Resolve incompatible source values.	Declared values must occur in the relevant corpus.
Constraint intersection	Identify the candidate satisfying all constraints.	The answer and declared near-miss candidates must be present.
Multi-hop evidence	Combine distributed clues and eliminate distractors.	Target, features, and distractors must occur in the corpus.
Temporal reasoning	Apply the specified reference date.	Declared stale and current values must both be supported.
Ambiguous reference	Resolve a description among plausible entities.	The clue must not already name the answer; target and distractors must exist.
Unsupported subgoal	State that requested information is unavailable.	A declared missing probe must not be found by the implemented corpus matcher.

paths, solution approach, and reference files. Structured `twist_audit` records encode the claimed source conflict, candidate set, time-dependent values, or missing subgoal. The verifier tests necessary conditions for these claims against shipped and captured material. Table 5 summarizes the distinction between an intended reasoning challenge and what the checks can establish. These are construction mechanisms, not measured effects on model scores.

The mechanism vocabulary is shared across authoring, schema validation, and verification. Prompt instructions and semantic linting guide difficulty selection, but quality advice is not universally a blocking gate. Some optional authoring policies require explicit configuration. In particular, matching a near-miss entity does not prove that it violates exactly one constraint, and absence of a lexical probe is not proof of semantic unsolvability. Such claims require execution and independent assessment.

Rewriting with preservation constraints. Scratch generation authors from source directions; regeneration can restructure an existing task and rederive factual values; conversion treats supplied rubric intent as an alignment constraint. On a feedback round, the prompt distinguishes difficulty adjustment from corrective rewriting. A corrective instruction takes precedence while compatible scenario anchors and previously established requirements are carried forward. The new specification passes through rubric generation and emission again. Maintaining an earlier scenario is an instruction and provenance obligation, not a proven guarantee that a model preserves every semantic constraint.

Layered checks and their execution scope. Emission distinguishes structural errors from quality advice and an absent rubric. A package can be emitted with a missing rubric warning, but the verifier rejects it as ungradable. The normal batch build invokes the verifier after emission and can route selected failure categories to rebuilding. The calibration loop invokes grounding verification before acceptance or salvage, after the solver–judge measurements. Checks cover reachable materials, declared reference sources and values, supported leakage patterns, and declared semantic twists. Warnings expose unreadable formats and unsupported checks. Model-based rubric review and recomputation serve additional roles; a registered optional quality hook must not be reported as a gate on every construction run. The study records which checks actually executed for each version and requires independent assessment of final usability.

A.6 Agentic judging and score admission

Evidence access and tool use. The construction loop calls a mounted scoring entry point with the emitted item and solver run directory. The judge receives private rubric and reference bindings plus access to the produced workspace, tool events, and conversation. Its configured tools include document readers, spreadsheet comparison, trace and transcript readers, and optional visual or external-fact checks. The inspected judge tool surface has no general code executor or automatic computed-specification checker; declared metrics and artifact paths are context for inspection, while the construction loop has a separate recomputation guard. Tool availability does not establish that a particular ballot used a tool: saved tool calls and evidence anchors provide that record. A required

visual surface or process transcript can be unavailable, in which case the criterion is excluded rather than assigned an invented outcome.

Ballots and evidence-sensitive aggregation. In the batch route, one judge conversation can inspect evidence for multiple criteria; separate invocations supply ballots, without an independence guarantee. For a binary criterion c that reaches aggregation in solver run a, let $\mathcal{B}_c$ contain normalized returned records from non-spoiled ballots that include that criterion. Invocation errors spoil a ballot; omission of c is an abstention for c. Counts n_c^+ and n_c^- record pass and fail votes. Let $d_c^+ = 1$ when a passing record is labeled deep, and $d_c^- = 1$ when a failing record is labeled deep and has a nonblank evidence anchor. Otherwise these flags are zero. The aggregation rule used in Equation 3 is

$$\widehat{y}_{a,c} = \begin{cases} \perp, & |\mathcal{B}_c| = 0, \\ 0, & n_c^+ \le n_c^- \text{ or } (d_c^- = 1 \ \wedge \ d_c^+ = 0), \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

Thus ties fail, and a pass majority can be overridden by an anchored deep dissent when no passing record is deep. These labels and anchors are judge-reported; the rule does not independently validate their truth. An unavailable evidence surface can also exclude a criterion before voting. Criterion indices distinguish authored rubric entries even when they share a capability category. Comparative criteria instead use their configured ordinal aggregation. Explicit exclusions leave the weighted denominator; the implementation's zero-valued output when no graded weight remains is not evidence that the solver failed every criterion.

Completeness and the scoring contract. The judge's completeness summary tracks missing or failed grading, including budget, API, and no-verdict cases; exclusion for an unavailable evidence surface need not lower that summary's completeness. The construction wrapper requires a valid score in $[0, 1]$, no budget exhaustion or reported judge-incompleteness, full reported data completeness, and rejection of identifiable all-fallback output. Missing legacy path details cannot independently authenticate judge execution. These are checks on the scoring contract, not a guarantee that every rubric criterion was gradable. An admitted score may therefore cover fewer than all criteria. A complete calibration vector further requires an admitted score for every configured solver. Rubric coverage, verdict completeness, and vector availability are separate.

Judge outputs guide construction; the fixed-output controls in Appendix C separately probe criterion-specific scoring behavior.

A.7 Reproducing the distributed analyses

The anonymous archive contains a self-contained offline reproduction path for the capability matrix, the arithmetic reference check, and the deterministic controller checks. From the extracted archive root, install the declared Python dependencies and run the following commands:

```
python -m pip install -r requirements.txt
OPENBLAS_NUM_THREADS=1 python code/plot_capability.py
  -source-root .  -out-dir outputs
python verify.py
python diagnostic/verify_arithmetic.py
python -B controller_checks/reproduce.py -output
LOCAL_CHECKS.json
```

The wrapped plotting command is entered on one line. The plotting script reconstructs Figure 3 from frozen published scores and anonymized raw per-item scores; the verifier checks its numerical output against the distributed reference. The arithmetic script checks the declared duration sum, while the native judge observations are preserved as frozen results rather than regenerated offline. Controller replay compares all fixture observations with the recorded production-source results. These commands make no model calls or private-service requests; dependency installation may require network access. Renderer versions and PDF metadata may change figure bytes without changing the numerical comparisons.

The archive's `SHA256SUMS.json` records a digest for each distributed file except itself. This release reproduces the included analyses and controls; it does not ship the private task corpus or the

historical human-review records. Consequently, its offline checks should not be interpreted as a rerun of the full construction pipeline.

B Daily-Life Instantiation and Task Examples

B.1 Released collection: composition and selection

The daily-life application is an instantiation of Agent-AutoBench. Its source-processing, authoring, execution, and evaluation artifacts belong to the system described in this paper.

Release composition and provenance. The audited release contains 100 packages across 11 domains, each with metadata, a seed, an instruction, a rubric, reference files, and an identifier-matched saved authoring specification. Saved seeds record 39 scratch, 37 regeneration, and 24 conversion outputs. The public inventory preserves anonymous identifiers and artifact hashes.

Selection and measurement scope. We keep implementation snapshots, release cohorts, and execution versions explicit because measurements collected under different configurations should not be pooled merely because they share the system name. The release includes human-supplied materials and selection using observed model scores and domain quotas. Its size measures selected production outputs, not source-to-output yield or independently assessed usability. The capability matrix uses a separate earlier evaluation cohort with a different domain distribution.

B.2 Multi-city travel: task environment and version boundary

Saved artifacts and intended outputs. The multi-city travel task begins with booking materials, weather references, restaurant pointers, and incomplete user preferences. Its saved specification contains task instructions, a weighted rubric, evidence bindings, a reference solution, and a private user-simulator specification. The intended deliverables are an itinerary CSV and a Markdown report. Table 2 maps the task artifacts to the behaviors they make assessable; descriptions summarize the saved specification rather than claiming that every retained execution exercised every interface successfully.

Historical version mismatch. Selected solver–judge records are bound to the task-artifact hashes of their historical version. For one time-window value, the saved authoring specification records 739 minutes, while the retained emitted rubric records 737; its hash differs from that historical binding. The example describes the saved specification, and we do not attach historical scores to the changed rubric. The discrepancy is a versioned construction defect, not an accepted numeric target for this manuscript.

Relation to the reference-correction study. The fresh paired intervention in Appendix C.2 uses a separate frozen task and supplies the direct correction evidence; historical scores from this example are not used to measure that effect.

B.3 Six-day travel: archived requirements and evidence limits

Task requirements and translation. A distinct second travel-planning task, separate from the opening multi-city case, makes the package representation concrete. Its archived instruction and rubric specify a task in which the agent must reconcile fixed bookings, family-chat updates, unclear handwritten notes, and official attraction information when preparing a six-day itinerary and budget. Table 6 summarizes selected requirements and one criterion from that payload. The English text is a condensed translation; output names are described by role rather than substituted as literal English filenames.

Interpreting the evidence requirements. This example illustrates why instruction, evidence, and rubric must be checked together. The intended response to an unreadable note is to acknowledge the uncertainty; a confidently invented number violates the requirement even if the output files are well formed. Likewise, the budget total and narrative itinerary must use the same price-verification basis.

Table 6: **Archived requirements and criterion links in the six-day travel task.** Requirements and rubric entries illustrate how task artifacts depend on shared evidence.

Component	Archived content, condensed and translated
Task requirement	Respect fixed bookings and the latest family constraints. Do not infer budgets or dietary restrictions from illegible notes.
Source materials	Three travel-chat screenshots, two handwritten notes, and itinerary and booking information. The payload also lists four attraction reference pages.
Required outputs	A six-day itinerary document and a budget spreadsheet in `output/`; the spreadsheet has seven specified columns and a verifiable-total row.
Rubric structure	Thirteen weighted criteria, with total weight 118.
Criterion A10	Extract reliable information from images and decline to use low-confidence handwriting; weight 7.
A10 checks	Recover chat timing, speakers, and decisions; identify the notes' lack of reliable fields and units; avoid inventing a budget, dietary restriction, or schedule.
Linked criteria	B4 checks whether a venue's price is explicitly verifiable; B1 uses that basis for the total; G5 checks consistency across outputs; D7 requires evidence attribution.

The task's pricing rule is a specified conservative placeholder, not a claim that the resulting amount is the appropriate ticket purchase.

Runtime mapping and assessment limits. This example is not linked to the recovered repeat-score groups and does not establish independent usability or comparative system performance. The archived instruction requests official-page lookup, while the payload separately lists reference pages. The runtime mapping between those requests and available materials must be checked before reuse. The surviving excerpt alone does not establish that the environment implements every requirement, that the reference facts are correct, or that the item passed independent review. It serves as an artifact example; the excerpt has no joined solver-score panel or verified repair trajectory.

C Fixed-Output Controls for a Subsequent Scorer Version

C.1 Fixed-output scoring: protocol, completion, and evidence

Diagnostic population and frozen protocol. This separate diagnostic evaluates a subsequent scorer version, without an established configuration link to the historical endorsement audit, using 5 fixed output variants from 2 synthetic tasks, covering 8 capability instances with 3 ballots each. Inputs, rubrics, and expected contrasts were frozen before scoring; expected outcomes were withheld from the judge. Each variant received one scoring run under the fixed native stack and its existing visual-model fallback policy. Evidence generation and judge consumption were checked separately. The protocol stopped on missing ballots or evidence, source drift, or unexpected outcomes, without replacement. The 24 planned ballots share tasks, evidence, and judge configuration; they are not independent samples or a count of API calls.

Prepared output contrasts. The masked PDF preserves the normal report's extractable text while visually covering table data cells. Removing section markers instead preserves the body and tables. The HTML controls contrast an effective click handler with a no-op. Table 7 reports the original verdicts and incomplete ballots.

Table 7: **Fixed-output scorer controls.** Original native verdicts with pass/fail/unknown (P/F/U) ballots. Missing ballots remain unknown; repeated ballots are not independent samples.

Fixed output	Native verdict; P/F/U	Evidence and limitation
Normal PDF	J7 pass; 3/0/0 J8 pass; 3/0/0	Rendered pages; actual image consumption
Working HTML	J10 pass; 3/0/0	Actual prepared click, changed content; on-demand browse failed
Masked table cells	J7 fail; 0/3/0 J8 pass; 3/0/0	Text extractable; cells visually covered
Removed section markers	J7 pass; 3/0/0 J8 fail; 0/3/0	Headings/dividers/frames absent; body retained
No-operation HTML	J10 fail; 0/2/1	Actual click leaves content unchanged; final ballot timed out (protocol incomplete)

Completed and missing ballots. All 5 variants were attempted; 4 completed their ballot allocation and 1 remained partial. Of 24 planned ballots, 23 were valid: 15 pass and 8 fail; 1 remained unknown. For the no-operation HTML, the final invocation timed out without a valid ballot or SDK completion record. Its native aggregate was fail and its summary marked complete, but the protocol's ballot requirement was unmet. We retain both records. Neither the timeout's default false-valued error object nor visual observations returned during that invocation count as another ballot. No result was retried, rejudged, or imputed.

Evidence consumption and browsing failures. Actual prepared interaction evidence was available and used, but each HTML control also had 3 failed `browse_html` calls: synchronous Playwright was invoked inside an asynchronous context and misreported as `chromium_missing`. Some normal-HTML anchors also confused a step's index field with its array position. The observations therefore concern prepared evidence, not healthy on-demand browsing. Preliminary browser failures and `none`-backend placeholders remain separate from grading.

Reported model identities, cost coverage, and interpretation. The agent and visual tools requested `gpt-5.6-sol`; SDK responses disclosed that name, while visual responses disclosed `deployment-gpt-5.6-sol-platform-global`. These labels do not establish hidden backend equivalence. Visual-tool and timed-out invocation charges were unavailable, so full-stack cost is unknown. These task-dependent controls establish neither general scoring accuracy nor repair or transfer benefit, and do not authenticate the historical audit's scorer configuration.

C.2 Selected arithmetic correction: a paired reference-target intervention

Case selection and the corrected target. This exploratory follow-up uses 1 package selected after an earlier arithmetic failure was observed. The earlier verdicts only motivated selection; every cell in Table 3 is a newly collected native judgment. Integer recomputation of the explicit visit durations gives $120 + 3 \times 60 + 4 \times 20 = 380$ minutes. The intervention changes the rubric's expected total and the equivalent private reference duration from 320 to 380 minutes. All other package files, task instructions, evidence, and both answer artifacts remain byte-identical. This is a deterministic reference correction, not an independently assessed autonomous constructor repair.

Frozen conditions and judge configuration. The protocol and input hashes were frozen before scoring. Conditions cross original versus corrected targets with fixed incorrect versus correct outputs, in a prespecified shuffled order. Each receives 1 native batch ballot, using the same Agent-as-Judge implementation, `single_judge_v1` profile, and requested `gpt-5.6-sol` model; caches, retries, and auxiliary visual-model calls are disabled in every condition. The judge receives the normal package and answer, without experimental labels or expected verdicts. The endpoint is criterion B1, not the correctness of every output claim or the overall weighted score. Other criteria can vary across fresh judgments.

Scheduling amendment and completed comparison. The initial preparation attempt exceeded its setup window before any ballot started; a documented scheduling amendment opened a fresh window without changing the scientific design. The completed comparison uses only the new window. All 4 arithmetic cells completed and the input hashes remained unchanged. The anonymous supplement contains the criterion outcomes, source and protocol digests, and an offline arithmetic verification script.

Within-case interpretation. One selected package and one ballot per condition provide a within-case diagnostic, with no sampling confidence interval, repeatability estimate, or population accuracy claim. Exact hidden backend identity is not established by the requested model label.

C.3 Deterministic retrieval and disclosure checks

Frozen fixtures and production entry points. A behavioral protocol with synthetic evidence and user facts was frozen before invoking the original production functions. The main and Daily-Life snapshots were tested through `SearchServer.do_search/do_fetch` and `ControlledUserSimulator.decide/commit`, without a language-model backend, online retrieval, or monkeypatching of controller logic. Table 8 reports every planned case. Each result is an assertion about the specified fixture, not a population accuracy estimate or a human assessment.

Table 8: **Direct checks of production information-access functions.** The same frozen fixtures exercise both implementation snapshots.

Predeclared behavior	Main	Daily-Life
Exact query returns captured evidence	Pass	Pass
Unrelated query returns no fabricated result	Pass	Pass
Fetch returns the captured plain-text content	Pass	Pass
Opening decision exposes the designated public fact	Pass	Pass
Matching clarification releases the authorized fact	Pass	Pass
Unrelated clarification releases no additional fact	Pass	Pass
Never-disclose fact remains outside disclosed state	Pass	Pass
Committed fact is not issued again on the repeated query	Pass	Pass

Source provenance, replay, and the packaging dependency. The supplement contains the exact fixtures, assertions, original observations, and source-derived offline replay. Anonymous source copies remove comments and docstrings and reformat the AST while retaining executable statements; original and distributed source digests are both preserved. Replaying these copies reproduces every original observation and predicate. The first snapshot replay encountered an import dependency for the Daily-Life disclosure module; its unrun cases are retained. Supplying the existing configuration

override for an unused model label resolves that packaging dependency without changing the protocol or tested logic.

Coverage limits. The unrelated-query test uses a short sequence: later stall-recovery policies can volunteer facts. Fetch equality is tested on plain text, not arbitrary HTML bytes. Paraphrase robustness, semantic fallback, rendered language-model replies, long conversations, and task-level usability are outside this diagnostic.

D HISTORICAL STATISTICAL PANELS

These analyses concern different historical populations and are not linked as a single causal sequence. Table 9 distinguishes the cohorts and their observation units. The archive's capability scores, repeated runs, construction outcomes, costs, and reviews were collected for different purposes and do not form a single controlled experiment.

Table 9: **Cohorts and units of observation.** These records describe the same Daily-Life application, but their denominators and version evidence differ.

Cohort	Unit and size	Selection and available linkage
Versioned release	100 packages	Score and domain-quota selection; frozen membership and package artifacts.
Capability panel	100 items, 24 solvers	Published aggregates and raw pairwise overlap; no established release-version join.
Signature-matched repeats	305 item–solver–round groups	Adaptive repeats; criterion IDs and weights match, full versions are unverified.
Recovered repeat panel	16 score lists	Recovered from a committed vector figure; item and tier identities unavailable.
Gate outcomes	344 records	Retention outcomes; score-to-final-version consistency is unverified.
Platform endorsements	7,815 criterion decisions	Non-blind submitted-work agreement; no package-usability denominator.

D.1 CAPABILITY SCORES AND RANK SENSITIVITY

The matrix in Figure 3 reports published subset scores; Figure 5 uses raw per-item means throughout. Solver identifiers remain fixed across the two displays according to the original published ordering, even when the raw-score ordering differs.

Published scores and raw paired comparisons. Figure 3 orders solvers by the published aggregate and subsets by their mean score. Cell labels are percentages on a normalized color scale from 0 to 1; subset sizes are uneven. The matrix reads the exported `score` field directly; exports containing `score_revise_revised` receive an outline. The supplement retains both `score_revise_origin` and the revised value, without estimating an adjustment formula. The platform's adjustment rule is not recoverable from these exports. Paired comparisons therefore use raw pairwise common items (subset-specific overlap ranges in the public manifest; 93–100 pooled). The nominal sign-flip tests require sign exchangeability under the null. Subset tests enumerate all sign assignments; pooled tests use 131,072 Monte Carlo assignments with seed 20270118. Rejection shares are descriptive, without multiple-comparison correction.

D.2 REPEATED SCORES: GROUPING, PROVENANCE, AND RESAMPLING

Signature-matched archival repeats. The analysis in Section 3 groups scored runs by item identifier, solver, and normalized round tag, after excluding runs flagged for API errors or incomplete judging. It retains groups sharing the multiset of graded capability identifiers and weights. This signature does not authenticate instruction, rubric text, environment, or execution configuration. Repetition was adaptive, several groups can share an item, and selection favored unstable verdicts. These restrictions apply to both the observed score ranges and their scalar-threshold crossing fraction.

Recovered repeat panel: provenance and selection. The separate repeat experiment sampled from a curated corpus with greater representation near the scalar threshold. The surviving analysis further depends on which executions produced sufficiently complete repeated scores. After the original scored-run archive became unavailable, the 16 group-level score lists were recovered from a committed vector figure. Item, tier, and stratum identities cannot be reconstructed.

Empirical resampling results. This separately initiated repeat experiment has a surviving panel of 16 recovered score lists and 52 scores, with 25.0% of lists crossing the scalar threshold. Figure 6

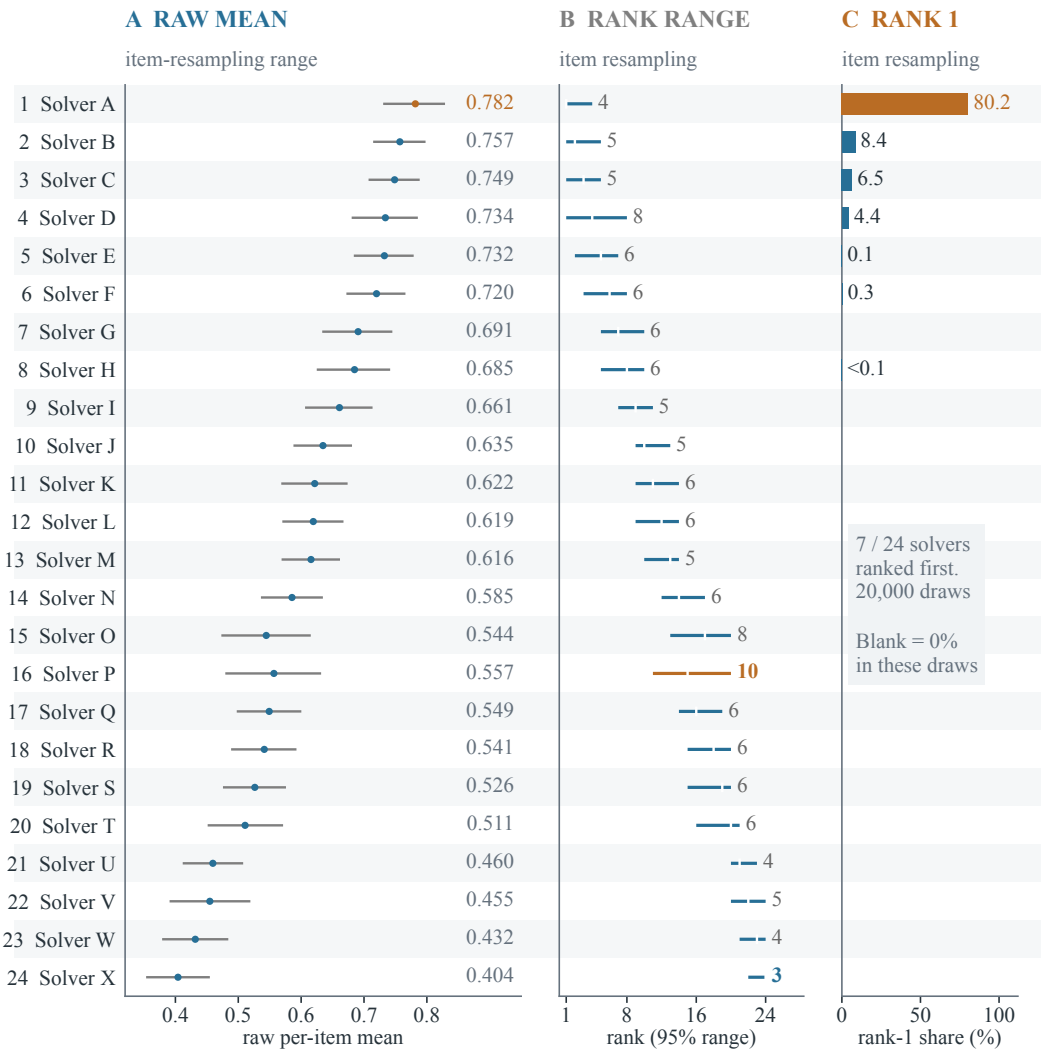

Figure 5: **Historical ordering under item resampling.** All score and bootstrap panels use raw per-item values with platform sentinel rows excluded. Panel A shows means and the 2.5th–97.5th percentile range of item-resampled means; panel B shows the corresponding rank range rounded to integer ranks, with white marks for observed raw-mean ranks and numbers for inclusive range widths. Panel C shows rank-one frequencies; blank entries have zero observed frequency. Alternating row shading is a reading guide. Missing observations remain missing; each resampled mean uses that solver's available sampled items. For each of 20,000 non-stratified resamples (seed 20270118), the same 100 item indices are drawn jointly for all solvers, holding recorded model–item scores fixed. Exact score ties follow the fixed solver-label order. Rank intervals and rank-one frequencies describe sensitivity to resampling the observed item mixture. They are neither probabilities of a model being truly best nor estimates of variability from rerunning the same fixed tasks. The published platform aggregates, including their adjustments, are shown separately in Figure 3.

preserves every recovered score and the frozen resampling calculation. Comparing median-threshold decisions from two independent batches sampled with replacement from each group's observed scores gives 11.2% disagreement for single-score batches and 8.3% for five-score batches.

Scope of the recovered-panel analysis. Larger batches here reuse observed values; they are not additional solver executions. The panel is selected, small, and lacks recoverable item identities, so it supports a sensitivity analysis of the observed scores rather than a population noise floor or a recommendation for the number of fresh runs. The frozen empirical resampling calculation is retained as a descriptive analysis of those lists; it cannot identify a population repetition budget or the effect of collecting fresh runs.

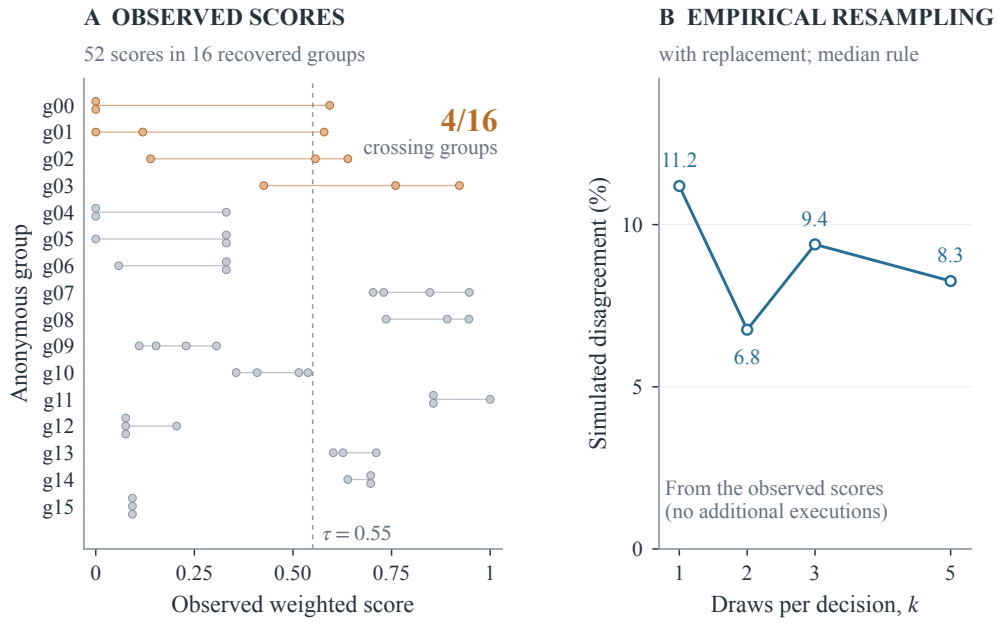

Figure 6: **Surviving observed scores and empirical resampling.** Panel A shows the recovered score lists relative to the scalar threshold $\tau = 0.55$; panel B shows disagreement under empirical resampling. The 16 recovered groups contain 52 scores; duplicate values are offset vertically. Original item, tier, and stratum identities are unavailable. For each group and k, the frozen calculation draws 4,000 pairs of batches with replacement from the observed scores and compares their median-threshold decisions (seed 20270118). Disagreement estimates are averaged equally across groups. Larger k denotes more draws from existing scores, not fresh executions, and does not establish a real-world sampling ceiling.

D.3 Retention outcomes, round coverage, and judging expenditure

Recorded outcomes and missing round counts. The gate-outcome archive contains 344 records: 94 are marked accepted, corresponding to 27.3%, and 243 carry the salvage outcome, corresponding to 70.6%. Salvage denotes retention with a posterior difficulty label rather than a recorded gate pass. Figure 7 shows which outcome groups retain round counts. Score-to-version consistency is not independently established for this archive. Both recorded outcomes precede independent validity assessment. Round counts are recoverable for 239 records, all from the salvage group; accepted records lack recoverable round counts. The archive therefore does not identify a rounds-to-acceptance distribution or the benefit of increasing the budget.

Recorded judging expenditure and missing cost joins. The grading archive records total judging expenditure of $19,890. Among 3,789 runs with positive recorded cost, the median is $4.39 per run. Solver execution and item generation are not separately metered in these figures. An item-level join between expenditure and gate outcomes is unavailable. These records support archive expenditure

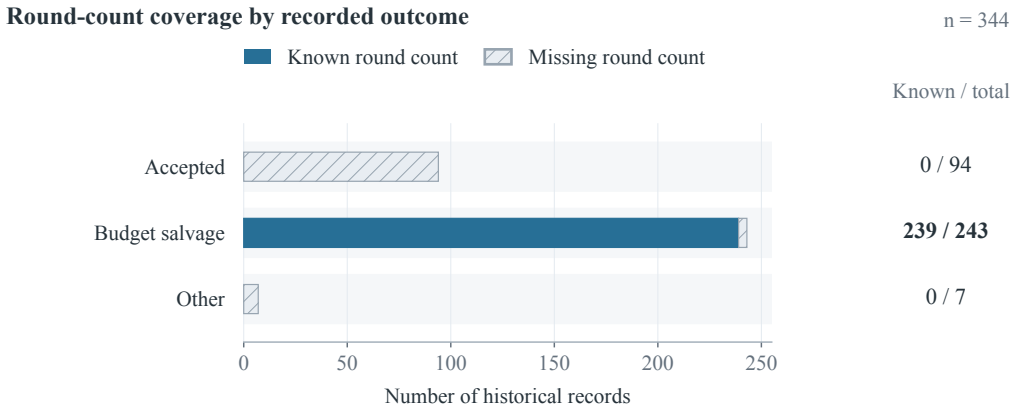

Figure 7: **Archival round-count coverage.** All accepted records lack recoverable round counts; the recoverable counts belong to the budget-salvage group. This outcome-dependent missingness prevents the observed round histogram from being interpreted as a rounds-to-acceptance distribution. Counts are computed directly from the frozen gate-outcome records.

and per-run summaries; estimating cost per usable output requires complete construction metering and independent item decisions in the same cohort.

D.4 SEPTEMBER 20 SNAPSHOT: NON-BLIND REVIEWER ENDORSEMENT

Snapshot selection and observation units. The anonymized platform snapshot was retrieved on September 20, 2026; the latest Daily-Life review was submitted on August 27, 2026. It is an expanded historical audit, not a new annotation experiment. We retain all platform project groups in the Daily-Life application series, including every named version, irrespective of agreement or platform QA status. The resulting 7,815 criterion reviews cover 216 distinct item identifiers, 300 platform tasks, 10 solver labels, and 9 reviewer accounts. The unfiltered export contains 8,338 reviews across applications. Repeated items across tasks remain associated within item clusters; these units are not interchangeable independent sample counts.

Review procedure and interpretation. Reviewers saw machine judgments; the export provides agreement responses rather than independent human pass/fail verdicts. Thus the result measures non-blind endorsement, not blind judge accuracy, human–human reliability, or package usability. Historical model labels and batches do not authenticate one common current judge configuration. Table 10 separates endorsement rates by machine verdict.

Table 10: **Criterion-level human endorsement in the September 20 snapshot.** "Missing" means the exported machine-verdict field is absent, although an agreement decision exists. "Excluded" denotes a non-scored criterion.

Machine verdict	Reviewed	Endorsed	Contested	Agreement (%)
Pass	3,017	2,800	217	92.81
Fail	4,686	4,629	57	98.78
Excluded	102	102	0	100.00
Missing	10	8	2	80.00
All	7,815	7,539	276	96.47
Pass/fail only	7,703	7,429	274	96.44

QA sensitivity and snapshot separation. The primary denominator includes all retained decisions. Platform-QA-approved records contain 6,501 endorsements among 6,758 reviews (96.20%); QA-unreviewed records contain 1,037 of 1,055 (98.29%), and QA-rejected records contain 1 of 2 (50.00%). QA approval is platform metadata, not an independently elicited pass/fail label. Restricting this snapshot to the project names used in the July snapshot yields 2,904 endorsements among 2,981 decisions (97.42%); the snapshots are retained separately and are not pooled or interpreted as a trend.

Weighting and item-cluster uncertainty. Equal-item and equal-reviewer averages are 96.95% and 96.53%. For the main-text interval, we use 20,000 item-cluster resamples with replacement (seed 20260920), retaining every associated criterion, task, solver, and reviewer record. Each resample recomputes the pooled endorsement ratio; the central 95% percentile interval conditions on the observed archive and fixed reviewers. The public script preserves the original export's item order and the frozen random-number procedure.

D.5 JULY 30 SNAPSHOT: EARLIER NON-BLIND ENDORSEMENT AUDIT

Snapshot selection and observation units. We recompute agreement from the frozen platform export dated July 30, 2026, using the four explicitly named Daily-Life project groups. This retains 2,939 criterion-level reviews and excludes unrelated application records; the unfiltered archive contains 2,942 reviews with 2,867 endorsements (97.45%). The application-specific cohort spans 150 distinct item identifiers, 152 platform tasks, and 210 task–item–solver groups. These are different units; platform groups are not independent task counts. Project labels identify historical batches, not a verified common judge-model, prompt, rubric, or task-version configuration.

Review procedure and endorsement definition. A reviewer sees the machine judgment and records agreement or disagreement; each disagreement in the export has a written reason. We define agreement as the number of endorsed criterion decisions divided by the number of reviewed criterion decisions. The export contains no independently elicited human pass/fail verdicts. We therefore

report endorsement rates, without inferring blind classification accuracy, chance-corrected agreement, or inter-rater reliability. This is evidence about archived criterion judgments, not a validated single current scorer configuration or independently assessed package usability. Table 11 reports this earlier snapshot separately.

Table 11: **Human endorsement by machine verdict in the Daily-Life audit.** All retained reviews have an agreement decision. "Excluded" denotes a non-scored criterion. Contested pass and fail decisions indicate different directions of disagreement, without assuming that the human response is an independent ground-truth verdict.

Machine verdict	Reviewed	Endorsed	Contested	Agreement (%)
Pass	1,154	1,108	46	96.01
Fail	1,683	1,654	29	98.28
Excluded	102	102	0	100.00
All	2,939	2,864	75	97.45
Pass/fail only	2,837	2,762	75	97.36

Weighting and uncertainty. The primary rate weights reviewed criteria equally. Equal-item and equal-reviewer averages are 97.49% and 97.51%, respectively; individual reviewer rates range from 96.33% to 98.16%. For this earlier snapshot, we draw 20,000 bootstrap samples of item identifiers with replacement (seed 20260917), retaining all associated criteria, tasks, solver labels, and reviewer records within each sampled item. Each replicate recomputes the pooled ratio; the central 95% percentile range is $[96.42, 98.26]\%$. This quantifies sensitivity to resampling the observed item mixture, conditional on the fixed reviewers.

Selection and annotation limits. Non-random review selection, exposure to machine decisions, shared reviewers, and unavailable first-pass revision histories limit extrapolation to new items or reviewers. No matched criterion has multiple independent reviewers in the retained export. Independent blinded annotation and human–human agreement would strengthen this audit, but cannot be recovered by relabeling the existing endorsement records.

D.6 RECORDED SOLVER CONFIGURATIONS

Table 12 connects the fixed solver labels in the performance panels to public model names recorded in the archive. Figure 3 displays these names directly; the fixed IDs also identify rows in the raw-score resampling panel. References identify the corresponding model through its developer's official model card, documentation, or release announcement. They do not identify the evaluation's API provider or authenticate an exact serving checkpoint. Configuration suffixes are recorded archive labels, not additional model names. Unavailable identities retain their fixed solver labels; no newer model is substituted for a historical configuration.

Table 12: **Public model names and recorded configuration labels.** IDs match Figure 3. Scores are the published 100-item aggregates, including platform adjustments, not the raw means in the resampling panel. A dash denotes no suffix in the public label. References point to the model developers.

ID	Recorded model / reference	Suffix	Score (%)
A	Anonymized configuration	—	78.17
B	Anonymized configuration	—	75.70
C	GPT-5.6 Sol (OpenAI, 2026a)	`max`	74.87
D	Grok 4.5 (SpaceXAI, 2026)	—	73.40
E	Anonymized configuration	—	73.13
F	GPT-5.5 (OpenAI, 2026b)	`xhigh`	71.99
G	Anonymized configuration	—	68.98
H	GPT-5.6 Terra (OpenAI, 2026c)	`max`	68.49
I	Anonymized configuration	—	66.09
J	Gemini 3.6 Flash (Google, 2026a)	`high`	63.47
K	Anonymized configuration	—	62.16
L	GPT-5.6 Luna (OpenAI, 2026d)	`max`	61.93
M	MiniMax M3 (MiniMax, 2026)	—	61.57
N	Kimi K2.7 Code (Moonshot AI, 2026a)	—	58.55
O	Anonymized configuration	—	56.66
P	Anonymized configuration	—	55.67
Q	DeepSeek V4 Pro (DeepSeek-AI, 2026)	`max`	54.92
R	Kimi K2.6 (Moonshot AI, 2026b)	—	54.14
S	Qwen3.7 Plus (Alibaba Cloud, 2026)	`high`	52.63
T	Gemini 3.5 Flash (Google, 2026b)	`high`	51.15
U	Gemini 3.1 Pro (Google DeepMind, 2026)	`high`	45.96
V	Qwen3.7 Max (Qwen Team, 2026)	`high`	45.48
W	Gemini 3.5 Flash Lite (Google, 2026c)	`high`	43.16
X	Anonymized configuration	—	40.41