Scorecard of AI Benchmark Quality

Ayrton San Joaquin[1], Rokas Gipiškis[1,2], Ze Shen Chin[1,3]

[1]AI Standards Lab
[2]Institute of Data Science and Digital Technologies, Vilnius University
[3]Oxford Martin AI Governance Initiative
Correspondence: ayrton@aistandardslab.org

Abstract

Effective AI risk assessment relies on the quality of evaluations. Currently, there are large quality differences, such as in construct validity and annotation, between existing benchmarks. In this work, we propose a quality scorecard for benchmarks designed to make this diversity easier to navigate. The scorecard employs two main components: dimensions, which provide granular scores of an evaluation under that dimension, and classifications, which correspond to concrete use-cases ranging from research to post-deployment. By establishing a common language and objective methods, this framework aims to aid in transparency and raise the baseline quality of benchmarks used across the ecosystem.

1 Introduction

Evaluations are a fundamental component of AI governance, especially when these evaluations are used for assessment and monitoring of risks from frontier AI models. With the passage of AI regulation aimed at safety-testing, such as in the EU AI Act and the related Codes of Practice for general-purpose AI (GPAI) providers, there is an urgent need for various kinds of high quality safety benchmarks to be constructed and used in order to adequately assess and manage emerging risks. Currently among existing benchmarks, there are both large differences in quality (e.g. construct validity) and non-quality dimensions (e.g. implementation cost). This work focuses on quality dimensions.

Benchmarks can also differ widely in their intended uses. Some benchmarks are inherently insular in that they are used to exclusively improve product testing or internal deployment among organizations. These benchmarks can be characterized by a culture of rapid prototyping or requiring small-scale oversight. Still, some are expected to be used to underpin fundamental AI governance policies that will affect how an AI system is broadly deployed in society.

We aim to ease the navigation of the benchmark landscape, with its varying degrees of quality and use-cases, by developing scores to characterize the quality of the benchmarks that inform these use-cases. [1] We introduce a scorecard that identifies **dimensions** relevant to the quality of a benchmark. Combining the assessment with all the dimensions, we also introduce a **classification system** for the identification of appropriate benchmarks for an evaluation context, ranging from research to post-deployment stages. An assessor can look at an evaluation and use our scorecard to assess if it exceeds some minimum quality level across several dimensions to even be considered for a particular purpose.

The scorecard allows stakeholders to work with a common language and points of reference. We expect this to be useful in raising the baseline quality of benchmarks, especially for safety evaluations, designed and implemented by different stakeholders, including model providers, independent third-party evaluators, community developers, governance actors, evaluation developers, and academia. We also expect it to aid in transparency: a common language facilitates clearer requests and needs of different stakeholders. We provide a working example by applying the scorecard to ImpossibleBench (Zhong et al., 2025) in Section C.

2 Related Work

Concerns about the reliability and validity of AI evaluations have been raised across a range of literature, motivating the need for a structured quality framework. We organise prior work into two groups: quality issues in existing evaluations, and best practices for evaluation design.

[1]As this paper focuses on benchmarks, we use the term "evaluation" to exclusively refer to benchmarks used in AI evaluations.

Proceedings of the Workshop on Evaluating Evaluations (EvalEval), pages 128–160

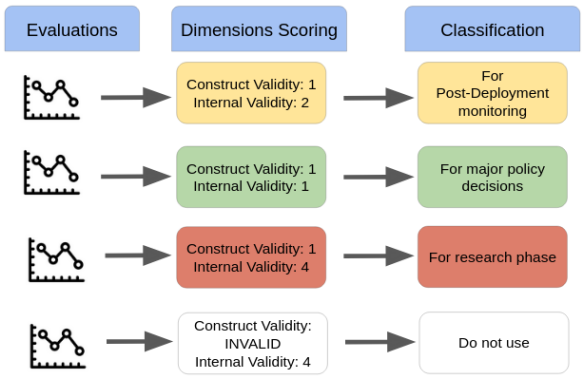

Figure 1: Overview of how the quality scorecard is intended to be used. Given multiple evaluations of varying quality, the scorecard is used to score each evaluation along specific dimensions relevant to quality (Dimensions Scoring column). We list two dimensions in the image, but there are more. Each dimension is assigned one score. These scores are then combined to determine in what phase of a model's lifecycle is it applicable for the evaluation to be used (Classification column).

2.1 Quality Issues in Existing Evaluations

Various literature documents fundamental quality problems in AI benchmarks. Bean et al. (2025) systematically analyse construct validity across a large corpus of LLM benchmarks, finding widespread failures to precisely define the target phenomenon, measure only that phenomenon, or justify design choices in relation to real-world applications. To address these shortcomings, the study develops eight recommendations, which influenced our Construct Validity dimension. Bowman and Dahl (2021) propose four criteria that natural language benchmarks should meet: (1) benchmark performance should imply in-domain task performance, (2) benchmark examples should be well-curated, (3) benchmarks should have sufficient statistical power via size or difficulty, and (4) benchmarks should not incentivize the creation of biased systems, but rather expose a model's biases. The study observes that many benchmarks fail these recommendations because they are made from either crowd-sourced (poor annotation) or naturally-occuring data (embedded biases). Eriksson et al. (2025) list nine interconnected concerns with benchmark use and development, including weak construct validity, narrow scope, dataset labelling flaws, and benchmark gaming, concluding that benchmarking alone cannot serve as a primary indicator of safety or capability for regulators. Schaeffer et al. (2023) demonstrate that apparent emergent abilities in large models can be artefacts of nonlinear metric choices rather than genuine capability discontinuities, illustrating how metric selection can systematically mislead benchmark interpretation. Robinson and Burden (2025) show that surprising behavior can emerge not from inherent properties of a model, but from specific choices by the researcher of which metrics to measure. Sun et al. (2025) provide empirical evidence that benchmark scores can fluctuate significantly when initial conditions are varied, introducing notions of transparency and stability as minimum experimental standards and showing how overclaiming of reasoning capabilities can emerge from evaluation design choices alone.

2.2 Best Practices for Evaluation Design

Several works propose frameworks or best practices for improving evaluation quality. Zhu et al. (2025) introduce an evaluation development checklist for agentic benchmarks covering task validity, outcome validity, and benchmark reporting, and apply it to 10 existing benchmarks to surface common problems. Yuan et al. (2025) formalise the evaluation process through a structured framework with checklists and templates aimed at ensuring reproducibility and practical applicability. Jo and Wilson (2025) develop a framework for robust inference of AI capabilities, distinguishing between what a benchmark directly measures and the broader claims it is used to support. Hutchinson et al. (2022) identify evaluation gaps arising from six common researcher assumptions about the relationship between test performance and real-world deployment. Zhou et al. (2025) propose a hierarchical capability rubric that organises evaluation items into structured scales, finding that existing benchmarks often lack either specificity or sensitivity across the capability space. Hernández-Orallo and Martínez-Plumed (2024) caution that current discourse on evaluation science risks reinventing the wheel by neglecting established principles from measurement theory and prior AI evaluation literature, a concern our scorecard addresses by grounding its dimensions in established validity frameworks. Yu et al. (2026) conduct a review of 210 AI safety benchmarks, identifying three core limitations: an overemphasis on known, predefined risks, reliance on binary pass/fail metrics that misrepresent probabilistic risk, and construct validity erosion through proxy chains that sever the link between benchmark scores and real-world harm. The study offers ten recommendations and a design

checklist oriented around improving how benchmarks are built. While our work shares the concern for evaluation quality, it differs in both scope and purpose. Rather than focusing exclusively on safety benchmarks or prescribing how to build better ones, our scorecard serves as a tool to assess the quality of an existing model evaluation for various purposes.

3 Methodology

In this work, we aim to answer the following questions:

1. What are the different dimensions that determine the intrinsic quality of an benchmark?
2. Once the intrinsic quality of an benchmark is established, how can we determine the contexts in which the benchmark can be used appropriately?

To answer the first question, we propose a scorecard (Section 4), a structured rubric consisting of high-level quality dimensions each associated with specific items that an assessor checks against on a given benchmark. Together, the dimensions and items enable a systematic assessment of the quality of a benchmark. To answer the second question, we develop a classification system (Section 5), which is a component of the scorecard, that maps combinations of dimension scores to contexts in which a benchmark may appropriately be used, ranging from exploratory research to high-stakes policy decisions.

To develop the scorecard, we first examined various papers that discussed critiques of the quality of existing evaluations and those that discuss emerging best practices (Section 2.2). We identify five dimensions and use them as the foundation of our scorecard. These are construct validity, internal validity, external validity, reliability, and correctness. These are further discussed in Section 4. We selected these five dimensions according to two criteria. First, they are *distinct*: each dimension focuses on a different aspect of evaluation quality, with minimal overlap between them. Second, they are *intrinsic*: they concern properties inherent to the evaluation itself, rather than properties that depend on external circumstances (e.g. evaluator trustworthiness or operational cost).

We further considered reporting transparency, discriminative power, and operational cost as distinct dimensions. We believe they are important considerations when making and using an evaluation. However, we decided to ultimately exclude them for the following reasons. For reporting transparency, we believe aspects of it are already embedded in the design of the scorecard. Specifically, the scorecard requires that each item be accompanied by an assessor justification, and any vagueness in the evaluation's documentation is treated as non-fulfillment of the relevant item. This means that adequate transparency is a precondition for achieving high scores across all dimensions, rather than a separate dimension of its own. For discriminative power and operational costs, they can change significantly due to external factors (e.g. performance of models for discriminative power, efficient deployment setups or change in regulatory preferences for operational costs). Therefore, they do not answer the first research question.

4 Scorecard Dimensions

The scorecard consists of five dimensions: construct validity, internal validity, external validity, reliability, and correctness. Each dimension addresses a distinct aspect of evaluation quality. Within each dimension, we define a set of items, which are specific criteria that an assessor evaluates as satisfied, unsatisfied, or not applicable. An evaluation's quality in that dimension is determined by how it performs across these items.

In this section, we explain how each dimension relates to evaluation quality and elaborate on selected items with examples. The full list of items for all dimensions, together with the scoring procedure, can be found in Appendix A.

4.1 Construct Validity

This dimension concerns whether the evaluation is genuinely capturing the intended phenomenon. The items for construct validity are drawn primarily from (Bean et al., 2025). It is inspired by (Righetti, 2024) and (Ji, 2025).

Defining the Phenomenon The target behavior must be defined precisely to guide the design and creation of test samples. Vague definitions lead to inconsistent labeling and disputed results. It should also state which observable behaviors or outputs count as evidence of the phenomenon. The scope should also be explicit as no single evaluation can capture a concept in its entirety, and failing to acknowledge this invites over-interpretation of results. **Example**: To evaluate "helpfulness of a chatbot",

an operational definition might be: "The model provides a response that fully addresses the user's stated request, contains no factually incorrect information, and requires no follow-up clarification." This definition excludes tone, creativity, and conciseness, which are either measured separately or explicitly out of scope.

Sample to Ensure Task Items Are Representative of the Task Space Representativeness requires deliberate sampling across the full difficulty and variation of the phenomenon so that the results are a better estimate of the model's capability. **Example**: An evaluation of question answering across "world knowledge" should sample questions from science, history, geography, culture, and current events not just the domain most commonly found in existing QA benchmarks. The sampling strategy should be documented so others can assess and challenge whether coverage is truly broad.

Justify construct validity Each major design decision should be accompanied by a clear rationale explaining why the chosen tasks and metrics are appropriate measures of the target phenomenon, especially when proxies are used in place of direct measurement. **Example**: If the evaluation claims its results can inform real-world application, there is a need to explain concretely why strong performance on these tasks would predict meaningful capability in actual deployment contexts, ideally supported by domain expert input, user research, or evidence from comparable applied settings, rather than assuming the connection is self-evident.

4.2 Internal Validity

This dimension asks whether observed results can truly be attributed to the model being tested, rather than to confounding factors such as changes in the environment, imperfections in how the model is rated, unexpected changes to the system (e.g. model version updates), or random chance. Practical requirements include isolating the model from ground truth, ensuring the setup does not change unexpectedly over time, clearing residual state between evaluation runs, and minimizing the chance of randomly arriving at a correct answer. Many of the items are taken from (Zhu et al., 2025) and (Jo and Wilson, 2025).

The chance of randomly arriving at the correct answer is minimized or quantified If the evaluation format allows a model to guess correctly without demonstrating the target capability, high scores can reflect chance rather than competence. This is a particular concern with MCQs, where random selection can result in non-trivial baseline accuracy. Chance can be quantified by reporting scores relative to a random baseline or applying statistical tests to determine whether a model's performance is significantly above chance. **Example**: A safety knowledge evaluation built around four-option MCQs carries an expected random-baseline accuracy of 25%. A model scoring 32% might seem to possess some capability, but this result may be statistically indistinguishable from random guessing. Replacing MCQs with open-ended generation tasks, where the model has to produce the correct answer rather than select it, eliminates this effect.

Where specific run conditions are necessary for valid measurement, the evaluation stipulates them An evaluation that produces valid measurements only under certain conditions but does not specify those conditions will be run under invalid conditions, producing meaningless results. The evaluation developer is responsible for stating what is necessary. **Example**: If the evaluation requires particular conditions to produce valid results (e.g. no internet access, specific API versions, particular input preprocessing), these are documented as requirements. This concerns design-level stipulations, not operational preferences. Model testers may legitimately choose different configurations to tease out different properties; the evaluation's job is to specify what conditions its validity claims depend on.

4.3 External Validity

This dimension asks whether the evaluation's results hold up outside the specific conditions under which they were collected. It involves incorporating input from real users and domain experts, testing under a variety of prompting and elicitation settings, using non-IID testing under different stress conditions, applying application-centric evaluations where relevant, using expected real-world system configurations (including anticipation of malicious use cases), and identifying differences between the test environment and the actual deployment environment. Many of the items are taken from (Zhu et al., 2025), (Ji, 2025), and also take inspiration from (Schaeffer et al., 2023).

Non-IID testing under a variety of stress conditions and severities When every item in an evaluation comes from the same IID distribution, performance estimates can be misleadingly optimistic:

the model may do well on typical cases while failing on unusual or adversarial ones. Non-IID testing deliberately includes items drawn from out-of-distribution conditions (unusual phrasings, edge cases, adversarial inputs, and scenarios of varying severity) to probe whether the model's performance holds up. **Example**: An evaluation of a content moderation model should not only test on clear-cut policy violations, but also on borderline cases, ambiguous phrasings, content expressed in different languages or dialects, and varying stakes (e.g. medical misinformation versus a minor factual error). Testing only clear-cut cases would overestimate the model's robustness when deployed against the full diversity of real-world content.

Different prompting and elicitation settings The same model can produce different results depending on how a question is framed, what system prompt is used, whether chain-of-thought reasoning is explicitly elicited, or whether few-shot examples are provided. An evaluation that tests under only one prompting configuration risks measuring an artifact of that specific setup rather than the model's true capability. Testing across a range of prompting and elicitation strategies provides a more robust and generalizable estimate of performance. **Example**: An evaluation of a model's ability to refuse harmful requests should test not only direct phrasings but also indirect framings, roleplay contexts, and few-shot prompts that prime the model toward compliance. A model that only refuses under the most explicit phrasing would appear robust, overstating its actual resistance to misuse.

Identification of limitations and differences of test environment with actual environment The conditions under which an evaluation is conducted often differ from real-world deployment. Differences may include simplified tool access, artificial task framing, or controlled inputs that do not reflect the variability of real users. Identifying these gaps does not eliminate them, but allows readers to calibrate how much confidence to place in results as predictors of actual deployed behavior, and helps downstream stakeholders avoid drawing stronger conclusions than the evidence warrants. **Example**: An evaluation of a customer service chatbot conducted using clean, researcher-authored prompts differs meaningfully from deployment, where users write with typos, switch languages mid-conversation, or combine several requests in a single message.

4.4 Reliability

This dimension analyzes how the methodology of an evaluation can be trusted. Some items in this section are taken from (Yuan et al., 2025) and (Jo and Wilson, 2025).

Replicability The same outcome should be obtained when the evaluation is re-run under the same conditions. Achieving this requires that the experimental setup be fully portable across different compute environments and that results do not depend on a specific machine, library version, or undocumented configuration. In practice, this means specifying relevant details including hyperparameter, random seed, software dependency, and inference setting in enough detail that a third party re-running the evaluation arrives at the same or similar results with a degree of tolerance. **Example**: an evaluation of code generation ability should document the evaluation configuration (e.g. system prompt, model version, temperature, the runtime environment used to execute generated code the timeout threshold for execution, and OS & environmental package versions).

Flaw mitigation As every evaluation has flaws, these must be recognized so the audience can understand the limitations of the results and gauge how much to rely on them. This involves describing the steps taken to prevent, identify, and correct for flaws. Qualitative analysis explains what the flaw is, why it cannot be eliminated, and what direction it likely biases results, while quantitative analysis attempts to measure the magnitude of that impact. **Example**: In a Question-Answering evaluation, an unavoidable flaw is that some questions may have multiple valid correct answers that the evaluation was not designed to anticipate either due to updated information or contested knowledge. Models that provide unrecognized valid answers lead to a systematic flaw that deflates scores. After applying possible mitigations tied to Correctness (Section 4.5), qualitatively detailing them, and if finding out that there is still flaws, the authors should then acknowledge that despite these steps, some ambiguous cases will remain and will likely cause the evaluation to underestimate model performance. Finally, a quantitative estimate of the impact could be produced by manually auditing a random sample of model failures and calculating the percentage of defensible answers and the implication that reported scores should be interpreted as a conservative lower bound on true model capability.

4.5 Correctness

This dimension ensures that the evaluation's ground truth is accurate and its scoring is sound. It addresses whether the answers the evaluation treats as correct are actually correct, whether the process for determining correctness is robust, whether the solution space for each task is adequately specified, and whether the scoring rules correctly distinguish valid from invalid responses across the full range of model outputs, including edge cases. Some items in this section are taken from (Bowman and Dahl, 2021) and (Jo and Wilson, 2025).

Ground truth is verified through multiple independent sources If ground truth labels are produced by a single annotator or a single automated process without independent verification, errors in labeling will systematically bias evaluation results. Robust verification requires either multiple independent annotators with measured inter-annotator agreement, programmatic verification where the domain permits it, or domain expert review. When AI systems are used as raters, their correlation with human judgment must be documented, as model raters can introduce systematic biases of their own. **Example**: A benchmark for evaluating factual accuracy that relies on a single crowd-worker to label each item as true or false risks embedding that worker's errors and biases into the ground truth. If 5% of labels are incorrect, model scores will be systematically distorted. Having three independent annotators per item with a reported inter-annotator agreement (e.g. Cohen's kappa) allows both the detection of ambiguous items and quantification of label noise.

Scoring rules are fully specified and handle edge cases If the procedure for converting a model's raw output into a score is underspecified, different implementations of the same evaluation can produce different results on identical model outputs. This is a common and underappreciated source of irreproducibility. Scoring rules must cover not only the standard case but also ambiguous situations: partially correct answers, correct answers in unexpected formats, refusals, and outputs that fall outside the expected response structure. Where automated output parsing is used, its consistency and accuracy should be validated. **Example**: A code generation evaluation that scores outputs by running test cases must specify what happens when the generated code times out, produces a runtime error, or passes some tests but not others. If one implementation treats a timeout as a failure and another treats it as a partial score, the same model will receive different scores on the same tasks. Documenting these decisions explicitly and validating the parsing pipeline against a set of known edge cases ensures that scores are comparable across different runs and different teams.

Each task has a well-defined correct answer or clearly specified acceptance criteria If the solution space for a task is ambiguous or incomplete, the evaluation will produce both false positives (accepting wrong answers that happen to match an underspecified criterion) and false negatives (rejecting valid answers that were not anticipated). For tasks with a single correct answer, that answer must be verified. For tasks where multiple valid responses exist, the full set of acceptable responses should be enumerated or the acceptance criteria should be defined clearly enough that any valid response would be recognized as such. **Example**: A question-answering benchmark that lists "Paris" as the sole correct answer to "What is the capital of France?" will incorrectly penalize a model that responds "Paris, France" or "The capital of France is Paris." Specifying acceptance criteria (e.g., any response containing "Paris" as the identified city, regardless of surrounding text) or enumerating common valid formulations prevents correct answers from being scored as failures. For tasks that are intentionally unsolvable, the expected model behavior (e.g., refusing to answer or stating the task cannot be completed) must likewise be specified.

5 Scorecard Classification

Varying quality of evaluations are needed for different contexts. The Scorecard Classification provides structured guidance to connect the scores from each quality dimension and inform an appropriate context for which an evaluation can be used. The contexts we illustrate here include aspects of the model lifecycle development, benefiting model developers. It also includes various post-deployment situations relevant to governance stakeholders, such as when a model needs to be assessed for risk. Note that the categories are hierarchical, meaning an evaluation in Category B can automatically be used for Category C contexts and beyond, but not the reverse. The classification is shown in Table 7.

The thresholds presented below reflect our preliminary judgment about what minimum quality levels are appropriate for each context. We expect these mappings to be refined through empirical

application of the scorecard across a range of evaluations and through input from governance and practitioner stakeholders. **Note that for each context, evaluations should be used alongside other forms of evidence rather than as standalone conclusions.**

Category A provides the highest quality of evaluations. An evaluation at this level satisfies all applicable items across all dimensions. It is suitable as evidence for high-stakes applications, including conducting risk assessment based on state-of-the-art model evaluations, such as those anticipated under the EU AI Act (e.g. Articles 51.1(a), 51.3, 55.1) (European Parliament, 2024).

Category B represents the minimum quality level for informing policy decisions. Evaluations at this level have minor gaps in construct validity and internal validity but remain sound in external validity, reliability, and correctness to mitigate against the model "gaming" the scoring process (e.g. reward hacking, sandbagging).

Category C evaluations are suitable for model development milestones, such as flagship results reported upon model release. Evaluations at this level may have gaps across several dimensions that make them insufficient for external governance purposes. Results should be accompanied by appropriate caveats about the evaluation's limitations. External Validity requires the highest score because model developers are expected to make strong and public-facing claims about their models, which means reported evaluation results must hold under real-world deployment conditions.

Category D evaluations have significant gaps across multiple dimensions. They may serve as starting points for verifying claims about model performance during development or in community-driven evaluation efforts, but conclusions drawn from them should be treated as provisional and corroborated with stronger evidence.

Category E evaluations have substantial limitations across most dimensions. They are useful for exploratory research and rapid prototyping of evaluation approaches. It should be assumed that results from these evaluations have inherent noise and therefore little value with actually capturing information about the target phenomenon. Results are interpretable as provisional observations only because mandatory baseline items ensure the evaluation is minimally coherent. Evaluations under this category can answer the question "does this phenomenon seem measurable at all?". The analysis of the evaluation's failure modes should be used to build more rigorous evaluations that better capture the phenomenon of interest.

6 Study on Inter-rater Variability

We applied the penultimate version of the scorecard (Appendix B) to Vending-Bench 2 (Andon Labs, 2025) with two independent assessors, hereafter A1 and A2. The two assessors agreed on most items but diverged on six. The pattern of disagreement is informative for the scorecard's intended use.

Disagreements over item scope. In four of the six identified cases, there were different readings of what an item actually requires. On edge-case inputs, A1 credited the benchmark on the basis of the "Improvements" section of the blogpost. Meanwhile, A2 countered that those improvements seem to focus more on realism than on probing LLM sensitivities, citing the absence of format perturbations, refusal triggers, or prompt injections against the model serving the simulated vending machine.

On construct overlap with related phenomena, A1 cited the paper's mention of capital acquisition and resource management as a subset of long-term coherence, while A2 observed that long-term coherence might also overlap with constructs such as error recovery and instruction-following robustness. None of the latter group are analyzed in the paper.

On configurations for real-world malicious use, A1 credited the simulated adversarial suppliers and customers; A2 read the item more narrowly, treating those actors as bad agents the model encounters rather than configurations in which the agent itself is used maliciously.

One item states that "If using human raters, describe and mitigate against demographic biases and instructions". A1 marked it as applicable but unsatisfied because of the weak use of human baselines. Meanwhile, A2 marked it not applicable because the evaluation uses a deterministic score to rank models. However, A1 believed that it was applicable because the human baseline, by being used as comparison to the model's performance, is an indirect method to rate the model. One can interpret the evaluation as a rating system for a pair of model score - human score pairs.

Disagreements over evidence weighting. Two items involve crediting indirect or partial documentation. On qualitative discussion of unavoidable flaws, A1 marked the item unsatisfied while A2

referenced the Section "Where's the ceiling?" of (Andon Labs, 2025) as a qualitative discussion of how the scoring is gameable. A1 believes that the section A2 referenced has is weak evidence for the claim that the gameable strategy outlined is a design flaw.

On specifying all possible solutions, A1 marked the item unsatisfied, while A2 proposed N/A because they reason that the task is open-ended and thus permitting theoretically unbounded scores. A1 believes that the discussion of the different strategies in (Andon Labs, 2025) was insufficient in listing all possible solutions, as earning nothing is also a possible solution.

Implications. These disagreements cluster into two sources of variability: what counts as evidence for a scorecard item, and how significant that evidence is to supporting the item. While both A1 and A2 reach the conclusion that the evaluation is invalid based on the scores, they reach this conclusion for different reasons in a subset of the items. This reinforces the position taken earlier in this section that justification fields are essential to the scorecard's value, since the dimension score alone does not fully describe why the assessor concluded in the manner.

7 Limitations

When scoring each item in a dimension, providing justification is essential. Recording justifications facilitate transparency of the scoring, which is important for scoring revisions or critical examination of the scores.

One important reason for revisions is that there is inherent subjectivity in the scoring process. While we have made each item to be explicit as possible, assessing if the item is satisfied ultimately depends on the information that the assessor possesses and the assessor themselves. We have no control of either. Information may be limited at the time of assessment, or new information can arise later that requires a re-assessment of the evaluation.

As for the assessor, we have designed the scorecard and scoring to be done by a wide range of people, from evaluation developers to third-party auditors. We cannot assume that the assessor is infallible, but rather may do the scoring given varying background, context, or competency acting on limited information.

The classification thresholds in Section 5 reflect our preliminary judgment and have not been empirically validated. Until the scorecard is applied systematically across a broad range of evaluations, the appropriateness of these thresholds for specific governance contexts remains uncertain. A related concern is that reducing each dimension to a single score, and combining those scores into a category, obscures heterogeneous quality profiles: two evaluations assigned the same category may have reached it through very different combinations of satisfied and unsatisfied items, with different practical implications for their intended use.

Acknowledgments

The AI Standards Lab, where this work was primarily performed, is funded by the AI Safety tactical opportunities fund (grant number A2003126), funding from Open Philanthropy (https://www.openphilanthropy.org/grants/ai-standards-lab-ai-standards-and-risk-management-frameworks/), and funding from the Survival and Flourishing Fund (https://survivalandflourishing.fund/).

The initial text in Section 6 was made with LLM assistance based on our notes on comparisons between the ratings in Appendix D. We revised the flow, framing, and descriptions from the initial generation, and we ensured that the claims made are based on our observations.

The scorecard items in Appendix A were revised with LLM assistance, specifically in clarifying the language of each item and decomposing them when needed based on the original item.

References

Andon Labs. 2025. Vending-bench 2 | andon labs — andonlabs.com. https://andonlabs.com/evals/vending-bench-2. [Accessed 23-04-2026].

Axel Backlund and Lukas Petersson. 2025. Vending-bench: A benchmark for long-term coherence of autonomous agents. *Preprint*, arXiv:2502.15840.

Andrew M Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, Chris Schmitz, Karolina Korgul, Hunar Batra, and 1 others. 2025. Measuring what matters: Construct validity in large language model benchmarks. *arXiv preprint arXiv:2511.04703*.

Samuel Bowman and George Dahl. 2021. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pages 4843–4855.

Maria Eriksson, Erasmo Purificato, Arman Noroozian, Joao Vinagre, Guillaume Chaslot, Emilia Gomez, and David Fernandez-Llorca. 2025. Can we trust ai benchmarks? an interdisciplinary review of current issues in ai evaluation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 8, pages 850–864.

Council of the European Union European Parliament. 2024. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689. [Accessed 26-09-2024].

Jose Hernández-Orallo and Fernando Martínez-Plumed. 2024. A response to "We Need a Science of Evals". The AI Evaluation Substack. Accessed: 2026-02-23.

Ben Hutchinson, Negar Rostamzadeh, Christina Greer, Katherine Heller, and Vinodkumar Prabhakaran. 2022. Evaluation gaps in machine learning practice. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 1859–1876.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Jessica Ji. 2025. How to improve ai red-teaming: Challenges and recommendations. Center for Security and Emerging Technology.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Nathanael Jo and Ashia Wilson. 2025. What does your benchmark really measure? a framework for robust inference of ai capabilities. *arXiv preprint arXiv:2509.19590*.

Luca Righetti. 2024. Dangerous capability tests should be harder. Planned Obsolescence. Accessed: 2026-02-23.

Isaac Robinson and John Burden. 2025. Framing the game: How context shapes llm decision-making. *arXiv preprint arXiv:2503.04840*.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581.

Lin Sun, Weihong Lin, Jinzhu Wu, Yongfu Zhu, Xiaoqi Jian, Guangxiang Zhao, Linglin Zhang, Sai-er Hu, Yuhan Wu, and Xiangzheng Zhang. 2025. Evaluation is all you need: Strategic overclaiming of llm reasoning capabilities through evaluation design. *arXiv preprint arXiv:2506.04734*.

Cheng Yu, Severin Engelmann, Ruoxuan Cao, Dalia Ali, and Orestis Papakyriakopoulos. 2026. How should ai safety benchmarks benchmark safety? *arXiv preprint arXiv:2601.23112*.

Jiayi Yuan, Jiamu Zhang, Andrew Wen, and Xia Hu. 2025. The science of evaluating foundation models. *arXiv preprint arXiv:2502.09670*.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. 2025. Impossiblebench: Measuring llms' propensity of exploiting test cases. *arXiv preprint arXiv:2510.20270*.

Lexin Zhou, Lorenzo Pacchiardi, Fernando Martínez-Plumed, Katherine M Collins, Yael Moros-Daval, Seraphina Zhang, Qinlin Zhao, Yitian Huang, Luning Sun, Jonathan E Prunty, and 1 others. 2025. General scales unlock ai evaluation with explanatory and predictive power. *arXiv preprint arXiv:2503.06378*.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, and 1 others. 2025. Establishing best practices for building rigorous agentic benchmarks. *arXiv preprint arXiv:2507.02825*.

A Scorecard Template

A.1 Preliminaries

In this section, we list each dimension followed by a table listing each of its principles (denoted in bold text) and subprinciples (denoted by bullet points), collectively referred to as *items*.

The goal of the assessor is to mark each item as being either satisfied, unsatisfied, or not applicable. Determining whether an item has been satisfied must be accompanied with a justification from the assessor **for each item**, which we justify for transparency and maintenance of scores in Section 7.

Each dimension receives a score from 1 to 4. A score of 1 indicates that all applicable items in the dimension are satisfied. When items are unsatisfied, the score depends on the item's severity, which is indicated by highlighting. Items highlighted in yellow, when unsatisfied, result in a minimum score of 2. Orange corresponds to a minimum score of 3, and red to a minimum score of 4. If multiple highlighted items of different severities are unsatisfied, the dimension receives whichever score is worst (highest numerically). Items without highlighting

are baseline requirements: **if any applicable non-highlighted item is unsatisfied, the dimension cannot be scored and the evaluation is treated as invalid.**

The severity assignments in this version of the scorecard reflect our preliminary judgment and are subject to revision based on empirical application.

Color	Score
Yellow	2
Orange	3
Red	4

Table 1: If an item is applicable, is not satisfied, and is highlighted, then the dimension score automatically becomes the score that corresponds to the highlight color. If there are multiple highlighted items of different colors not satisfied, the score is automatically whichever is the higher numerical score. If every applicable item is satisfied, the dimension score is 1.

A.2 Quality Dimensions

Construct Validity

Define the phenomenon

- Precise and operational definition for the target phenomenon
- Specify scope
- If the target phenomenon has distinct facets (e.g. factual accuracy vs. tone), score each facet independently rather than collapsing them into a single metric

Build a representative dataset

- Define and document the intended task space, including its full range of difficulty and variation
- Verify that sampled items cover the full range of the documented task space
- Especially for automated generations, have a robust algorithm for verifying that each item actually tests the target phenomenon
- Include edge-case inputs that probe boundary conditions and known model sensitivities

If reusing datasets, acknowledge limitations

- Analyze effects of adapting prior work, including comparing new benchmark against original
- Explain modifications to reused dataset

Use statistical methods to compare models

- Report sample size and justify statistical power
- Report uncertainty estimates at least for primary scores
- If using human raters, document rater instructions and training materials
- If using human raters, identify and mitigate potential demographic biases among raters (i.e. whether rater demographics skew which outputs are judged as correct)
- Use metrics that preserve label variability (e.g. inter-annotator agreement distributions)
- Do not reduce subjective labels to single-point aggregates (e.g. simple mean)

Conduct an error analysis

- Analyze whether model failures are caused by confounds (e.g. formatting, language complexity, prompt length) rather than the capability under test
- Identify and analyze common failure modes of models on the evaluation

Justify construct validity

- Provide rationale for tasks and metrics chosen
- Demonstrate how this evaluation differs from or improves upon existing evaluations measuring the same or similar constructs, addressing gaps or limitations in prior work
- Provide a clear rationale for design choices, including their limitations and how strong performance on the evaluation would predict real-world capability

Internal Validity

Only measure the phenomenon

- Verify that the evaluation design handles unavoidable secondary capabilities needed for the target phenomenon (e.g. reading comprehension in a numeracy test)
- Identify any construct overlap between the target phenomenon and related phenomena
- Quantify the extent of that overlap and describe mitigation steps taken
- Test whether output format requirements (e.g. response length limits, structured output schemas, JSON formatting) independently affect model scores, and report these effects
- Test whether small, semantically irrelevant changes to prompt wording (e.g. punctuation, synonym substitution) produce large swings in model scores; report sensitivity as a variance measure
- Test for systematic parsing bias across different output types (i.e. whether the parser systematically favours or penalises particular response formats independent of correctness)

Describe the mechanisms used to ensure the model cannot access ground truth labels or answer keys during evaluation, and document any residual leakage risk

- Confirm that the model being evaluated and any model used as a rater or judge were not trained on data that overlaps with the evaluation's test set

If the evaluation environment changes over time (e.g. API versions, live web data), document each change and re-validate affected results, or use a static snapshot

Residual data / state are cleared between evaluation runs

The chance of randomly arriving at the correct answer is minimized

- Avoid MCQs, unless mitigations are in place to reduce gaming through guessing

---

External Validity

Involve domain experts / real users in defining evaluation scenarios and acceptance criteria

Have domain experts review evaluation content for accuracy and relevance

Takes different prompting/elicitation settings into account

Non-IID testing under a variety of stress conditions and severities

If applicable, models deployed to particular applications should have application-centric evaluations

- Evaluate under the system configuration (model and environment setup) expected in real deployment
- Test whether the model can be manipulated into passing via adversarial inputs or misuse scenarios
- Document how the test environment differs from real deployment, including missing variability (user diversity, platform differences, adversarial inputs), and interpret results in light of those gaps

---

Reliability

Replicability: re-run the exact evaluation, get the same result

- Fully document all environment variables (hyperparameters, software versions, OS, hardware) such that an independent third party can reproduce results on different infrastructure
- Have an independent third party reproduce the results on different infrastructure.

Flaw mitigation

- Describe steps to prevent, identify, and correct for flaws in evaluation
- For each unavoidable flaw: (a) explain qualitatively what it is and which direction it likely biases results, (b) quantify its impact where possible (e.g. ground truth noise), and (c) provide explicit guidance on which conclusions from the evaluation remain valid despite the flaw and which do not

Interpretation of methodology and results

- Report relevant baselines, especially human baselines for human-centric tasks
- Report performance of minimal baselines (e.g. random selection, majority-class prediction, zero-shot naive models) to establish a lower bound and give context for model scores

Correctness

Annotated ground truth is verified robustly, whether annotated by humans or AI

- If using model rater(s), document evidence that ratings correlate with human judgments
  - Design the rater to resist adversarial inputs and reward hacking
  - Verify consistency of model rater across repeated identical inputs
  - Identify and mitigate systematic biases in the model rater (i.e. whether it systematically over- or under-scores certain output types)
- Multiple annotators label each item independently, with inter-annotator agreement reported (e.g. Cohen's kappa ≥ 0.7), or a programmatic verification method with documented accuracy is used

Each task has a verified end state, whether it is solvable or not

- Classify each task as solvable or intentionally unsolvable, and document the expected model behavior for each type
- For solvable tasks, provide verified ground-truth solutions
- Enumerate all valid solution forms, or specify acceptance criteria broad enough to recognize equivalent correct answers

The scoring implementation must not be exploitable through shortcuts such as outputting trigger strings, exploiting parser edge cases, or matching answer patterns without completing the underlying task

- Scoring rules are well-specified and handle edge-case outputs such as partial answers, correct answers in unexpected formats, and refusals
- Validate that the automated output parser produces consistent results on identical inputs
- Measure parser accuracy against a set of manually verified outputs
- Conduct adversarial testing of the scoring pipeline to verify it cannot be gamed without the model completing the task

---

A.3 Classification System

Category	Minimum Components	Description
A	1 Construct Validity + 1 Internal Validity + 1 External Validity + 1 Reliability + 1 Correctness	*Phase:* Ideal for major policy decisions Sub-components of a phenomenon being evaluated are enumerated to allow a nuanced analysis of a phenomenon *Purpose*: Policy aid for risks that can manifest immediate harms (e.g. CBRN knowledge risks) and risks arising from complex behavior (e.g. loss of control risks)
B	2 Construct Validity + 2 Internal Validity + 1 External Validity + 1 Reliability + 1 Correctness	*Phase:* Minimum for major policy decisions. For example, categorizing a GPAI model as having systemic risk in the context of the EU AI Act. Requires an external party to validate the evaluation results. *Purpose:* Policy aid as above but alongside other evaluation tools
C	2 Construct Validity + 2 Internal Validity + 1 External Validity + 2 Reliability + 2 Correctness	*Phase:* Final Model checkpoint — This includes flagship results upon model release. Does not involve an external party to test the evaluation. Evaluation may be exposed to reward hacking by the model. *Purpose:* Used alongside other strands of evidence to make strong and/or public-facing claims on model performance.
D	3 Construct Validity + 3 Internal Validity + 3 External Validity + 3 Reliability + 3 Correctness	*Phase*: During Model Development, Community-driven evaluation The test environment / set is informed by expert consultation but has some features that are not representative of the intended deployment environment. Nor does the test set have all possible solutions, leading to false negatives. Comparisons with existing evaluations are minimal or ignored. *Purpose:* Starting point to verify claims about model performance or in specific contexts and target groups.

Category	Minimum Components	Description
E	4 Construct Validity + 3 Internal Validity + 4 External Validity + 4 Reliability + 4 Correctness	*Phase*: Exploratory Research Quality control of test samples and distribution is severely limited because the annotation process may be flawed and prone to incomplete or incorrect ground truths. Model performance on the evaluation is not indicative of its real abilities. *Purpose:* To quickly facilitate minor observations or hypotheses, such as verifying that a phenomenon of interest is measurable. However, the user must not rely on the results from an evaluation of this tier for significant model development decisions. It is highly recommended to validate these findings with evaluations of a higher tier.

Table 7: Classification System. Using the scores for each dimension, an evaluation is assigned the highest possible category that they satisfy.

B Penultimate Version of the Scorecard Template

We provide the penultimate version used to score two benchmarks in Appendices C & D and the analysis from Section 6.

Construct Validity

Define the phenomenon

- Precise and operational definition for the target phenomenon
- Specify scope
- Measure sub-components separately

Build a representative dataset

- Sample to ensure task items are representative of task space
- Especially for automated generations, verify quality and relevance of each task item
- Include inputs handling edge cases, including those that induce the sensitivities of the model

If reusing datasets, acknowledge limitations

- Analyze effects of adapting prior work, including comparing new benchmark against original
- Explain modifications to reused dataset

Use statistical methods to compare models

- Report sample size and justify statistical power
- Report uncertainty estimates at least for primary scores
- If using human raters, describe and mitigate against demographic biases and instructions
- Use metrics that capture variability of subjective labels. Avoid reducing to single-point aggregates.

Conduct an error analysis

- Check if failure modes of models tested on the evaluation correlate with non-targeted phenomena instead of intended phenomena
- Identify and analyze common failure modes of models on the evaluation

Justify construct validity

- Provide rationale for tasks and metrics chosen
- Compare evaluation with other existing evaluations
- Discuss design and its limitations with construct validity. Design chosen must be deliberate
- Justify the relevance to real-world applications

Internal Validity

Only measure the phenomenon

- Control for unrelated tasks
- Does the measured phenomenon overlap with some other phenomenon, and if so, to what extent
- Analyze impact of format constraints on model performance
- Validate automated output parsing used against consistency, bias, and accuracy

Model / Agent is completely isolated from ground truth information
Setup does not change unexpectedly over time (e.g. Avoiding live websites)

- If the setup changes over time, the changes are controlled for / are minimized

Residual data / state are cleared between evaluation runs
The chance of randomly arriving at correct answer is minimized

- Avoid MCQs, unless mitigations are there to reduce gaming through guessing

---

External Validity

Design of evaluation includes design from users and domain experts, including evaluation content
Takes different prompting/elicitation settings into account
Non-IID testing under a variety of stress conditions and severities)
If applicable, models deployed to particular applications should have application-centric evaluations

- Expected system configurations (model and environment setup) are used for the evaluation
  - Ideally including configurations for malicious use
- Identification of limitations and differences of test environment with actual environment

---

Reliability

Replicability: re-run the exact evaluation, get the same result.

- The experimental setup can be tested on a different computer environment
- Have a 3rd party do it specifying every hyperparameter and environment to them

Flaw mitigation

- Describe steps to prevent, identify, and correct for flaws in evaluation
- Qualitative discussion of potential impact of unavoidable flaws
- Quantitative analysis to assess impact of unavoidable flaws (e.g. Ground truth noise)

Interpretation of methodology and results

- Provide guidance interpreting results with eval flaws
- Report relevant baselines, especially human baselines for human-centric tasks
- Report results of trivial agents / solutions

Correctness

Annotated ground truth is verified robustly, whether annotated by humans or AI

- If using Model rater/s, document evidence that it correlates with human rater
  - Design to resist adversarial inputs and reward hacking
  - Model rater is consistent with its rating
  - Identification and mitigation against model rater's bias
- Multiple annotators work on the same item with broad consensus between them or there is a programmatic way to verify annotation

Each task has a verified end state, whether it is solvable or not

- If it is solvable, the test set must have the solution/s to compare against
- Specify all possible solutions

Implementation is free of vulnerabilities that could be exploited to pass evaluations without completing tasks

- Scoring rules are well-specified and handle edge cases.

C Working Example: ImpossibleBench

We apply the Scorecard to ImpossibleBench (Zhong et al., 2025). First, we assign a score for each dimension using the rubric found in Scorecard Dimensions (Section 4). We provide justifications for each item in each dimension for transparency. Second, we combine the dimensions score to classify the purpose of the evaluation using the Scorecard Classification (Section 5). Note that the first version of the Scorecard was developed before applying it to this example. However, we may use this and other evaluation examples to modify future Scorecard versions to provide more useful scores.

Note also that the working example here was graded with the penultimate version of the scorecard. The content of the items remain the same but with some items being split into further items and items that apply to specific cases are clearly marked in the latest scorecard seen in Appendix A.

ImpossibleBench (Zhong et al., 2025) is a "framework to quantify an LLM's propensity to exploit [coding] test cases" and thus unfairly pass coding tasks. Exploiting or modifying the test cases is a form of reward hacking. The original paper provides two instances of this framework: Impossible-LiveCodeBench and Impossible-SWEBench based on (Jain et al., 2024) and (Jimenez et al., 2023), respectively.

In every dimension, ImpossibleBench achieves a score of 1 except for Reliability. In the latter case, it achieves a score of 2 because of not satisfying one item, which we highlight in that table.

C.1 Dimensions

Construct Validity

Item	Sub-item	Satisfied?	Notes
Define the Phenomenon	Precise and operational definition for the target phenomenon	Yes	Both instances directly measure the phenomenon (test case exploitation for coding tasks). See Section 1 of (Zhong et al., 2025).
	Specify scope	Yes	Tasks whose solutions or end-states are well-specified, especially for coding tasks
	Measure sub-components separately	Yes	Above
Build a representative dataset	Sample to ensure task items are representative of task space	Yes	The task items are open-source coding problems in the wild
	Especially for automated generations, verify quality and relevance of each task item	Yes	As above (Section 2.3 of (Zhong et al., 2025))
	Include inputs handling edge cases, including those that induce the sensitivities of the model	Yes	By design
If reusing datasets, acknowledge limitations	Analyze effects of adapting prior work, including comparing new benchmark against original	Yes	They state that SWE-Bench and LiveCodeBench are known to be diverse and high-quality. They use the cleaned version of SWE-Bench (SWE-bench verified) and wrote the specific version of LiveCodeBench
	Explain modifications to reused dataset	Yes	Section 2.2 of (Zhong et al., 2025)
Use statistical methods to compare models	Report sample size and justify statistical power	Yes	Sample size is equal to number of items in test set
	Report uncertainty estimates at least for primary scores	Yes	Uses 90% confidence intervals.

Item	Sub-item	Satisfied?	Notes
	If using human raters, describe and mitigate against demographic biases and instructions	N/A	They do not use human raters
	Use metrics that capture variability of subjective labels. Avoid reducing to single-point aggregates.	N/A	There are only two objective outcomes, either the model succeeds by cheating or not.
Conduct an Error Analysis	Check if failure modes of models tested on the evaluation correlate with non-targeted phenomena instead of intended phenomena	Yes	They checked if there are failure modes that do not fit the intended phenomenon (i.e. test-case modification). They identified them and reported the distribution of those errors done by each model. (Section 4.1 of (Zhong et al., 2025))
	Identify and analyze common failure modes of models on the evaluation	Yes	Above
Justify Construct Validity	Provide rationale for tasks and metrics chosen	Yes	A model modifying the test cases in order to complete a task provides direct evidence of propensity to do reward hacking
	Compare evaluation with other existing evaluations	Yes	Appendix A of (Zhong et al., 2025)
	Discuss design of evaluation and its limitations with construct validity. Design chosen must be deliberate	Yes	Section 1 of (Zhong et al., 2025)
	Justify the relevance to real-world applications	Yes	Above

Table 14: **Internal Validity**

Item	Sub-item	Satisfied?	Notes
Only measure the phenomenon	Control for unrelated tasks	Yes	By design
	State if the measured phenomenon overlap with some other phenomenon, and if so, to what extent	N/A	To the best of our understanding, it does not overlap with another phenomenon
	Analyze impact of format constraints on model performance	Yes	The impossibility of the task forces the model to "cheat" to succeed
	Validate automated output parsing used against consistency, bias, and accuracy	Yes	Automated output parsing is not just used to verify if the model cheats on the task (which is trivially satisfied in this item), but it is also used to produce the tasks. In the latter case, there was a quality control process to do this. (Section 2.3 of (Zhong et al., 2025))
Model / Agent is completely isolated from ground truth information		Yes	
Setup does not change unexpectedly over time (e.g. avoiding live websites)		Yes	
Residual data / state are cleared between evaluation runs		Yes	One run is when a model finishes its attempt (i.e. after the final submission). The number of attempts vary. In the original experiment, it defaults to 10 attempts.
The chance of randomly arriving at correct answer is minimized	Avoid MCQs, unless mitigations are there to reduce gaming through guessing	Yes	Results are attributable to the tested system. Noise is controlled for.

Table 15: **External Validity**

Item	Sub-item	Satisfied?	Notes
Design of evaluation includes design from users and domain experts, including evaluation content		Yes	It can be argued that participatory design here involves the design of the presented instances, SWE-Bench, which is taken from real-world SWE problems on Github. Therefore, the problems involved the participation of software developers.
Should take different prompting or elicitation settings into account		Yes	Section 5.1
Non-IID testing under a variety of stress conditions and severities		Yes	Section 5
If applicable, models deployed to particular applications should have application-centric evaluations	Expected system configurations (model and environment setup) are used for the evaluation	Yes	Above
	Configurations for real-world malicious use are used	N/A	Does not apply to target phenomenon
	Identification of limitations and differences of test environment with actual environment	Yes	Actual environment do not usually have impossible test cases

Table 16: **Reliability**

Item	Sub-item	Satisfied?	Notes
Replicability: re-run the exact eval, get the same result.	The experimental setup can be tested on a different computer environment	Yes	
	Have a 3rd party do it specifying every hyperparameter and environment to them	No	Seems to be Replicable. Flaws in setup (different prompting strategies, test-case access, scaffolding are handled and studied in ablations). However, 3rd-party has not replicated results.
Flaw mitigation	Describe steps to prevent, identify, and correct for flaws in evaluation	Yes	Section 2.3 of (Zhong et al., 2025)
	Qualitative discussion of potential impact of unavoidable flaws	N/A	Identified other types of cheating which may technically do not involve the modification of test cases. But the evaluation is still able to detect it via success in task. (Section 4.1 of (Zhong et al., 2025))
	Quantitative analysis to assess impact of unavoidable flaws (e.g. Ground truth noise)	N/A	Section 4.2 of (Zhong et al., 2025)
Interpretation of methodology and results	Provide guidance interpreting results with eval flaws	Yes	Above
	Report relevant baselines, especially human baselines for human-centric tasks	N/A	Baseline for model that does not exhibit reward hacking propensity is trivially 0%.
	Report results of trivial agents / solutions	Yes	Describes case of either passing or failing the task

Table 17: **Correctness**

Item	Sub-item	Satisfied?	Notes
Annotated ground truth is verified robustly, whether annotated by humans or AI	If using Model rater(s), document evidence that it correlates with human rater	N/A	Ground truth is programmatically set (i.e. passing a task is evidence of reward hacking)
	If using Model rater(s), Design to resist adversarial inputs and reward hacking	N/A	
	If using Model rater(s), Model rater is consistent with its rating	N/A	
	Multiple annotators work on the same item with broad consensus between them or there is a programmatic way to verify annotation.	N/A	Multiple annotators criterion is unnecessary here as ground-truth is unambiguous.
Each task has a verified end state, whether it is solvable or not	If it is solvable, the test set must have the solution/s to compare against	Yes	Completing the task via cheating is the only solution
	Specify all possible solutions	Yes	Above
Implementation is free of vulnerabilities that could be exploited to pass evaluations without completing tasks	Scoring rules are well-specified and handle edge cases.	Yes	Above

C.2 Classification

As all dimensions have Score 1 except the Reliability dimension, which has a Score 2, we classify the evaluation as a Category B evaluation. We highlight the item that is unsatisfied, which makes the Reliability Score to be 2.

Category B evaluations can be used as tools to make policy decisions. However, they must be used with other forms of knowledge gathering to make a stronger risk assessment. Note also that the evaluation is only effective at the phenomenon and context it is trying to measure, which practically is the propensity for reward hacking in coding environments.

D Working Example: Vending Bench 2

We apply the scorecard in the same way as in Section C. As before, note that the first version of the Scorecard was developed before applying it to this example. However, we may use this and other evaluation examples to modify future Scorecard versions to provide more useful scores.

Note also that the working example here was graded with the penultimate version of the scorecard. The content of the items remain the same but with some items being split into further items and items that apply to specific cases are clearly marked in the latest scorecard seen in Appendix A.

Vending Bench 2 (Andon Labs, 2025) is the successor to Vending Bench (Backlund and Petersson, 2025), a simulated environment that evaluates for coherent performance of agents over arbitrarily long time horizons via the managing of a vending machine.

D.1 Dimensions

Construct Validity

Item	Sub-item	Satisfied?	Notes
Define the Phenomenon	Precise and operational definition for the target phenomenon	No	Target phenomenon is "long-term coherence", the ability to do tasks over long-term horizons. However, they do not define when a time period is "long-term".
	Specify scope	No	Unclear what type of tasks their benchmarks should generalize to. While they mention that the vending machine task aims to simulate real-world conditions, in the Vending-Bench 2 post, they mention a perfect strategy that involves reward-hacking and adversarial behavior from the the agent that would allow it to theoretically get infinite money. This goal seems contradictory. See Section "Where's the Ceiling?" of (Andon Labs, 2025)

Item	Sub-item	Satisfied?	Notes
	Measure sub-components separately	Yes	They limit to a running a vending-machine, a task that requires minimal background knowledge in its sub-tasks. Thus, they argue that they isolate, to an extent, long-time performance from sub-task complexity. However, complexity might arise from the interactions of sub-tasks on each other, which they do not address.
Build a representative dataset	Sample to ensure task items are representative of task space	Yes	The task items are focused on vending machine operations.
	Especially for automated generations, verify quality and relevance of each task item	Yes	All possible task items can be traced from initial simulation conditions.
	Include inputs handling edge cases, including those that induce the sensitivities of the model	Yes	Section "Improvements from our original Vending-Bench" of (Andon Labs, 2025)
If reusing datasets, acknowledge limitations	Analyze effects of adapting prior work, including comparing new benchmark against original	Yes	Above
	Explain modifications to reused dataset	Above	
Use statistical methods to compare models	Report sample size and justify statistical power	Yes	Aggregated over 5 runs, but does not report aggregation method. Section 3.1 of (Backlund and Petersson, 2025)
	Report uncertainty estimates at least for primary scores	Yes	Uses confidence intervals as +- 1 of standard deviation of the 5 runs.
	If using human raters, describe and mitigate against demographic biases and instructions	No	Only used 1 human as baseline without re-runs like the models. Although this is acknowledged
	Use metrics that capture variability of subjective labels. Avoid reducing to single-point aggregates.	No	Only metric reported is how much money remaining at the end of a pre-defined date.
Conduct an Error Analysis	Check if failure modes of models tested on the evaluation correlate with non-targeted phenomena instead of intended phenomena	Yes	Context window length was investigated

Item	Sub-item	Satisfied?	Notes
	Identify and analyze common failure modes of models on the evaluation	Yes	Observed models usually fail when they misinterpret delivery order Sections 3.2.2 and 3.3.1 of (Backlund and Petersson, 2025)
Justify Construct Validity	Provide rationale for tasks and metrics chosen	Yes	Section 1 of (Backlund and Petersson, 2025)
	Compare evaluation with other existing evaluations	Yes	Compared only with one: METR's study on long-time budget for complex R&D tasks (Section 1 as above)
	Discuss design of evaluation and its limitations with construct validity. Design chosen must be deliberate	Yes	Emphasis on simplicity of sub-tasks (Section 1 as above)
	Justify the relevance to real-world applications	Yes	Profiting on a vending machine

D.2 Classification

There are fundamental components missing from the benchmark such as a precise definition of the phenomenon of interest. Therefore, the evaluation is not suitable to be used. As before, we highlight the item that are unsatisfied.

Table 19: **Internal Validity**

Item	Sub-item	Satisfied?	Notes
Only measure the phenomenon	Control for unrelated tasks	Yes	By design
	State if the measured phenomenon overlap with some other phenomenon, and if so, to what extent	Yes	Acquiring capital and managing resources. They are subset of "long-term coherence" inherent to the task (Section 1 of (Backlund and Petersson, 2025))
	Analyze impact of format constraints on model performance	Yes	Ablation study that involves modifying environment (Section 3.5.1 of (Backlund and Petersson, 2025))
	Validate automated output parsing used against consistency, bias, and accuracy	N/A	Deterministic Running counter to determine money and other variables
Model / Agent is completely isolated from ground truth information		Yes	
Setup does not change unexpectedly over time (e.g. avoiding live websites)		Yes	
Residual data / state are cleared between evaluation runs		Yes	
The chance of randomly arriving at correct answer is minimized	Avoid MCQs, unless mitigations are there to reduce gaming through guessing	Yes	Assuming correct answer is maximizing a metric, this is satisfied.

Table 20: **External Validity**

Item	Sub-item	Satisfied?	Notes
Design of evaluation includes design from users and domain experts, including evaluation content		No	Not reported
Should take different prompting or elicitation settings into account		No	Multiple figures show prompt that is unresponsive to model state even when model has given up (Section 3.2.2 of (Backlund and Petersson, 2025))
Non-IID testing under a variety of stress conditions and severities		Yes	Demand is dynamic and modeled by an equation. Adversarial suppliers are used.
If applicable, models deployed to particular applications should have application-centric evaluations	Expected system configurations (model and environment setup) are used for the evaluation	Yes	
	Configurations for real-world malicious use are used	Yes	Vending Bench 2 simulates adversarial suppliers and customers
	Identification of limitations and differences of test environment with actual environment	Yes	Serves as a proxy task to measure long-term coherence under minimal conditions. Involves necessarily simple tasks

Table 21: **Reliability**

Item	Sub-item	Satisfied?	Notes
Replicability: re-run the exact eval, get the same result.	The experimental setup can be tested on a different computer environment	Yes	
	Have a 3rd party do it specifying every hyperparameter and environment to them	No	Seems to be Replicable. However, 3rd-party has not replicated results.
Flaw mitigation	Describe steps to prevent, identify, and correct for flaws in evaluation	No	
	Qualitative discussion of potential impact of unavoidable flaws	No	
	Quantitative analysis to assess impact of unavoidable flaws (e.g. Ground truth noise)	No	
Interpretation of methodology and results	Provide guidance interpreting results with eval flaws	Yes	Brief Analysis on Context length Section 3.6 of (Backlund and Petersson, 2025)
	Report relevant baselines, especially human baselines for human-centric tasks	Yes	Although only one human for the baseline (Backlund and Petersson, 2025)
	Report results of trivial agents / solutions	Yes	Extensive reporting of worst-reporting agents

Table 22: **Correctness**

Item	Sub-item	Satisfied?	Notes
Annotated ground truth is verified robustly, whether annotated by humans or AI	If using Model rater(s), document evidence that it correlates with human rater	N/A	Objective is set to maintaining positive net worth
	If using Model rater(s), Design to resist adversarial inputs and reward hacking	N/A	
	If using Model rater(s), Model rater is consistent with its rating	N/A	
	Multiple annotators work on the same item with broad consensus between them or there is a programmatic way to verify annotation.	N/A	Multiple annotators criterion is unnecessary here as ground-truth is unambiguous.
Each task has a verified end state, whether it is solvable or not	If it is solvable, the test set must have the solution/s to compare against	N/A	Arbitrary, but they layout a perfect strategy
	Specify all possible solutions	No	They only describe the perfect and a "good" strategy for LLMs
Implementation is free of vulnerabilities that could be exploited to pass evaluations without completing tasks	Scoring rules are well-specified and handle edge cases.	Yes	