International Network for Advanced AI Measurement, Evaluation and Science Best Practice: Automated Evaluation of Large Language Models

Table of Contents

A. Introduction, purpose and scope.

A.1. Introduction and purpose

Advanced AI is being developed and deployed across borders, industries, and languages. To build trust and support confident adoption, we need shared ways to understand these systems.

That's why the International Network for Advanced AI Measurement, Evaluation and Science - formerly the International Network of AI Safety Institutes - has re-focused its work on strengthening the science that underpins AI evaluations. Established in November 2024, the Network brings together Australia, Canada, the European Union, France, Japan, Kenya, the Republic of Korea, Singapore, the United Kingdom and the United States to build internationally recognised approaches for measuring and evaluating advanced AI capabilities.

Third-party evaluators are crucial to supporting trusted adoption of advanced AI. Businesses and governments, in the Network and outside of it, will require high-quality measurement science, and evaluators, to assess how to best adopt AI. This best practice document, developed from the work of Network members, is aimed at supporting these third-party evaluators.

To do so, this document builds on best practice documentation that already exists, most notably including NIST AI 800-2. This Network document is intended to complement existing work and to provide more specific targeting to third-party evaluators, such as on capability elicitation.

This document is structured to follow the sequential steps evaluators should take,

- Section A introduces and outlines the scope of the paper.
- Section B covers evaluation objectives and provides guidance on how evaluators should select their evaluations.
- Section C outlines the process for implementing and running evaluations. Sub-sections cover designing the evaluation protocol and conducting capability elicitation experiments, writing the evaluation code, running the evaluation and tracking results, and debugging the evaluation.

A.2. Scope

AI evaluation is a broad and rapidly evolving field, encompassing a wide range of methods, objectives, and approaches. This document focuses on one important subset: automated evaluations of AI models.

Automated evaluations - where model responses are graded automatically, often by an LLM - are increasingly central to how organisations measure and compare AI capabilities. They offer scalability for large amounts of evaluation results, and consistency for applying similar evaluations to different models, making them a

practical foundation for third-party evaluation work. Automated evaluations also come with particular limitations, which will be addressed by this document.

Automated evaluations will likely be among the first type of evaluations run by new third-party evaluators. The Network has chosen to focus on automated evaluations to support current and prospective third-party evaluators in running these evaluations effectively. This document also focuses on evaluators using pre-existing evaluations.

Within this, the document covers considerations relevant to a range of automated evaluation formats. The automated evaluations covered in this best practice document are largely capability evaluations – tests designed to measure the performance of AI models against certain tasks.

Automated evaluations include:

- Multiple choice evaluations, which are question-answer evaluations which involve the model selecting between several potential answers.
- Open-ended evaluations, which have many possible correct responses, typically when the evaluation domain cannot be divided into discrete tasks, and/or it is difficult to define objective grading criteria or verification procedures. Testing "rubrics" are often used to assess the validity of model responses.

This document does not cover how evaluators should build their own evaluations. The interpretation of results is also not in scope. The following evaluations are also out of scope:

- Safeguard or "red-team testing", which involves adversarial probing of AI systems to identify vulnerabilities. While some "red-team" testing can be automated, such as refusal testing, these remain out of scope for this document.
- Agentic evaluations, which assess AI agent performance, including in tool-use and long-horizon task execution. While this document's practices apply to many agentic evaluations as well, such evaluations raise additional methodological considerations - including agent reliability, scaffolding design, and sandbox configuration - that are addressed in more depth elsewhere, including in NIST AI 800-2, METR's Example Protocol for AI Agent Evaluations, Rabanser et al.'s Towards a Science of AI Agent Reliability, and Zhu et al.'s Establishing Best Practices for Building Rigorous Agentic Benchmarks.
- Expert capability assessment, which requires domain-specific expertise to evaluate whether AI systems can perform specialised tasks at a professional level.
- Propensity evaluations, which look to measure how likely AI models or systems are to behave or respond in certain ways.
- Human uplift studies, which measure whether AI systems meaningfully enhance a user's ability to carry out tasks, relative to a "status quo", such as an internet

search. Guidance on designing rigorous uplift studies is provided in RAND's Preliminary suggestions for rigorous GPAI model evaluations.
- The emerging field of open-world evaluations. More detail is held in Kapoor et al., 2026.

These areas are important to the broader evaluation landscape and may be addressed in future publications by the Network. Many evaluators may find it useful to conduct automated evaluations in conjunction with these other forms of evaluations, utilising a 'layered' approach to testing AI models and systems. However, this document focuses directly on automated evaluations themselves, providing a foundation for the evaluations that third-party evaluators are likely to carry out at scale.

B. Defining evaluation objectives and selecting evaluations.

B.1. Defining evaluation objectives

Evaluations should be done for a well-defined reason. Often, this is to gather evidence to assess claims about AI capabilities in real-world settings. But in all cases evaluators need to specify, before developing or executing their evaluations, what they are evaluating for. References to this should be included in evaluation reporting. While complementary best practice documentation, such as NIST AI 800-2, goes into more detail, this document will cover the high-level steps that evaluators should consider when defining their evaluation objectives.

Evaluators should follow three key high-level steps when defining what and how to measure: establishing the overall evaluation objectives, identifying the measurement construct, and ensuring construct validity.

- The first step evaluators should take is to determine *why* they are doing an evaluation. The purpose of the evaluation shapes every decision that follows - from what benchmarks to use, to how results are scored and reported. Potential reasons include comparing the capabilities of different AI models or assessing the cyber misuse risk of an AI model.
- Next, evaluators should clearly set out *what* they are trying to measure. This is often known as the "measurement construct." A measurement construct is the specific concept or capability the evaluation is intended to measure - for example, a model's ability to answer graduate-level science questions. Defining the measurement construct precisely and in advance helps ensure that the evaluation remains focused, that its results can be interpreted clearly, and bounds the scope of the claims that can be made from the results. More specifically, evaluators should,
    - Provide a precise and operational definition for what is being measured.
    - Specify the scope of what is being covered and acknowledge any excluded aspects.

    - Identify if what is being measured has sub-components and ensure they are measured separately, or in such a way to not implicitly confound the target construct or skill.
- Evaluators should also consider whether the evaluation measures the intended construct[1]. More specifically, evaluators should,
    - Justify the relevance of the benchmark for the phenomenon with real-world applications.
    - Provide a clear rationale for the choice of tasks and metrics, connected to the operational definition of the phenomenon.
    - Compare similarities and differences between the benchmark and existing evaluations of similar phenomena. In this case, evaluators should consider using human or AI baselines to provide a point of comparison, depending on the purpose of the evaluation.
    - Discuss the limitations and design trade-offs of the benchmark concerning construct validity.
    - Consider whether the evaluation is intended to measure or predict a real-world outcome, or to measure an abstract concept.

Evaluators should also consider potential business and policy objectives, particularly when performing evaluations for private firms; for example, a hiring application might prioritise recall (to avoid missing strong candidates) or precision (to filter out underqualified ones) depending on the specific objectives of the business. (For more detail, refer to section 2.1.2 of Singapore IMDA's "Starter Kit for Testing LLM-Based Applications for Safety and Reliability".)

B.2. Select evaluations that meet objectives.

Third-party evaluators may utilise evaluations developed by other parties, including public benchmarks. When doing so, evaluators should ensure that the chosen evaluations reflect the objectives selected at the previous stage.

The first step for most evaluators will be to conduct a "survey" of public benchmarks. There are a significant number of benchmarks available publicly. Some AI evaluation software, including Inspect and Moonshot, contain a significant number of public benchmarks already installed. When considering potential evaluations, evaluators should clearly document what they expect each benchmark to measure, and how this measurement relates to their overall objectives. In this survey, third-party evaluators assessing 'frontier' models – that is, models that possess cutting-edge capabilities –

[1] This is often referred to as "construct validity": whether an evaluation measures the concept it purports to measure. A related concept is external validity: whether results obtained under test conditions generalise to real-world conditions. Notably, Appendix 3 of the Safety and Security Chapter of the EU's General-Purpose AI Code of Practice requires model evaluations to demonstrate internal validity, external validity, and reproducibility.

should pay special attention to assessing whether the potential evaluations are "saturated", meaning that evaluation performance is no longer a useful differentiator between highly-scoring models. A more detailed picture of how different evaluations suit different tasks is included in NIST AI 800-2.

Once this survey is completed, evaluators should undertake the following validation steps to ensure that their selected evaluations are well-suited to their objectives.

- Document precisely what each candidate benchmark claims to measure, and the accuracy of this claim based on available information about its construction, validation, and usage by others. Where possible, manually inspect test items. Relevant details include subject matter, intended difficulty, item format, and how responses are graded.
- Document whether the benchmark directly measures the construct of interest, is conceptually related to the evaluation objective, or predicts downstream outcomes of interest. Where the benchmark directly measures the construct, consider coverage across the space of relevant tasks and whether the item format reflects intended use cases. Where the benchmark is conceptually related rather than a direct measure, draw on use cases, threat models, and subject matter expertise to build evidence for the relationship.
- Consider whether the difficulty range of the benchmark is appropriate for the intended use of results. Benchmarks that are too easy or too hard are not useful for comparing models. Where the evaluation seeks to compare AI performance to alternative approaches, validated baseline measures (e.g., human performance) are beneficial and should themselves be statistically robust. These baselines are useful for grounding the metric (e.g. for saying whether scoring 10% on the evaluation is a strong or poor performance)
- Assess the quality of the benchmark. Greater diversity of test items reduces the risk that shared idiosyncrasies affect results and supports broader inferences. The number of items affects statistical power, which can be assessed through power analysis. Consider contamination risk, noting that systems trained on benchmark contents or formats may score higher than they would on unseen data. Contamination risk can be reduced by keeping benchmarks questions and/or answers private.
- Consider practical factors including the human labour and computational cost required, compatibility with a range of AI systems, and whether others have previously reported results on the same benchmark, which can be useful for validation and contextualisation. Mougan et al contains further discussion of prioritisation in resource-constrained contexts.

While automated evaluations offer scalability and consistency, evaluators should remain conscious of their limitations. Automating judgement can constrain what is measured well: the decision to automate can create selection pressure towards

constructs that are readily machine-scoreable, unintentionally privileging proxy measures over the construct of genuine interest. Evaluators should therefore treat the decision of whether to automate grading as an explicit step in the design process, considering whether automation supports their objective without distorting the identified construct. Where automated scoring is used, it should be validated against human judgements on a representative sample - one of the most important procedural differences when moving from human-graded to automated approaches. Automated evaluations are often best understood as complementary to manual methods and other forms of evaluation, and can be particularly valuable as a baseline or screening tool alongside periodic human review.

C. Implementing and running evaluations

C.1. Design the evaluation protocol and conduct capability elicitation experiments.

C.1.1 Design the evaluation protocol

When implementing and running evaluations, the first step for evaluators is to design the evaluation protocol: the full set of operational procedures carried out during an evaluation. This includes decisions about what items to include, how prompts are formatted, how models are queried, and how outputs are scored. Evaluators building or adapting evaluations should design their protocol to directly support the objectives defined in Section B. Three common design principles, guided by NIST AI 800-2, can help guide protocol decisions:

Comparability: the protocol should support meaningful comparison across models, versions, or time points. It is useful to distinguish between two aspects of comparability: the evaluation of outputs (scoring criteria, rubrics, graders) and, the generation of outputs (prompt format, inference settings).

- Evaluation of outputs should always be comparable - using the same scoring criteria, rubrics[2], and graders ensures that evaluators are comparing the outputs themselves rather than differences in how they are judged.
- Generation, however, is more nuanced: if the goal is to test whether a specific prompt works better for a specific model, then prompts and settings should be held constant; but in some cases, the goal is to understand what a model is capable of, which requires eliciting its best performance under the right conditions. Using a prompt that doesn't work well on a given model tells you little about what that model could do if prompted appropriately. How strictly generation should be standardised therefore depends on the goals of the evaluation - comparability is more or less important depending on what evaluators are trying to learn.

[2] For more detail on rubrics, please consult Pencharz, Flesch and Sandbrick.

External validity: the protocol should reflect, as far as is practical, the conditions under which the model will be used in practice. For example, if the evaluation is intended to assess a model's usefulness in a customer-facing application, the prompt format, interaction style, and task complexity should approximate real-world usage rather than highly artificial or contrived scenarios.

Cost control: evaluations consume computational resources and human time. Evaluators should consider the trade-off between thoroughness and cost when designing their protocol, for example, by selecting an appropriate number of test items and trials that balance statistical power against budget constraints. These issues are likely to be relevant for third-party evaluators, who will often be resource constrained. An open research questions remains on whether less compute-intensive version of evaluations are considered good practice.

Evaluators should then consider whether aspects of the protocol should be optimised iteratively. In some cases, evaluation objectives may require evaluators to refine aspects of their protocol. For example, an evaluator seeking to measure the upper bound of a model's capability on a particular task may need to adjust prompts across successive runs to reduce refusals or improve task comprehension. Similarly, scoring rubrics for open-ended evaluations may need to be refined based on early outputs to ensure they capture the range of model responses appropriately. Evaluators should also check for the bias and calibration of the grader. Where iterative optimisation is used, evaluators should document each iteration, including what was changed, why it was changed, and how the change affected results. This documentation supports transparency and allows others to assess whether the final protocol represents a fair test of the model's capabilities.

C.1.2 Conduct capability elicitation experiments

Capability elicitation, referred to as performance optimisation in NIST AI 800-2, is the process of unlocking or enhancing a model's abilities after it has been trained in order to measure its capabilities at a level relevant to the evaluation objective. Without effective capability elicitation, evaluations risk significantly misestimating what a model can do, either by underestimating capabilities that the model possesses but does not display under default settings, or by the elicitation process overfitting. For evaluations that seek to measure reliability or typical real-world performance, capability elicitation continues to be important. In such cases, evaluators must ensure that the performance they observe reflects how the model would actually perform in practice.

This section sets out practices that evaluators should follow when conducting capability elicitation in automated evaluations. It covers the experimental process for elicitation, the low-level settings that evaluators should consider, and key practices around task and scoring settings.

Capability elicitation is fundamentally an experimental process, in which evaluators iteratively adjust aspects of the evaluation protocol to better measure the capability of interest. While more detail on this section is held in NIST AI 800-2 and a UK AISI blogpost, some important practices include:

- Evaluators can iterate on prompts either manually or using LLM-based prompt optimisation. Relevant iterations may include refining task instructions to reduce ambiguity, adjusting formatting to match model expectations, or tuning inference-time settings such as reasoning effort. Each iteration should be documented, including what was changed, why it was changed, and how the change affected results.
- When evaluators iterate on prompts, inference settings, or other aspects of the protocol, they should do so using data that is distinct from the set used to report final results. Failure to observe this is one of the most common and consequential mistakes in capability elicitation. Where possible, evaluators should adopt a three-way split of the evaluation data (more detail is held in the UK AISI protocol for capability elicitation document). While it should be noted that the exact percentage of each set depends on a case-by-case basis, evaluators should, as a starting point, look at,
    - An exploratory set of a small number of items (e.g., 2-5), used for manual experimentation and initial qualitative analysis of model behaviour.
    - A tuning set, used for iterating over prompts, parameters, and other elicitation choices. This should typically be 70-80% of the items.[3]
    - An evaluation set, used only for the final reported evaluation after tuning is complete. This should typically be 20-30% of the items.
- These three sets should be disjoint, meaning that they should be composed of completely separate data. Evaluators should not look at the evaluation set during tuning. Where benchmarks do not come pre-split, evaluators should construct their own splits and document them clearly, including the rationale for how items were allocated across sets.
- Evaluators should maintain clear records of which settings were tried, which were discarded, and the rationale for the final configuration used in reported results. Capability elicitation generates a large number of intermediate results. Documentation of these intermediate results supports transparency and allows others to assess whether the final protocol represents a “fair” test of the model's capabilities.

A core part of capability elicitation is the adjustment of settings that influence how the tested model generates its outputs, sometimes referred to as inference settings. While

[3] It should be noted that these percentages are a rule-of-thumb. Typically, what matters to evaluators is the *absolute* size of the exploratory or tuning set. For example, if evaluators had a very large set of items, there would not be a need to have 70-80% of the data be in the tuning set.

more detail on this section is held in NIST AI 800-2 and a UK AISI blogpost, evaluators should pay attention to,

- Sampling parameters (temperature, top_p, top_k): Evaluators should document and justify choices with regard to these parameters. Evaluators should also note that variance in outputs persists even at temperature zero due to non-sampling sources of stochasticity (e.g., batching effects on the provider side).
- Reasoning effort: Evaluators should note that higher effort generally improves performance at increased cost, though the tradeoff is domain-dependent and can be particularly pronounced in domains such as advanced mathematics. Evaluators should report reasoning effort settings alongside results and should set these consistently with the evaluation objective. For evaluations scored with pass@k, evaluators should also be aware that the number of attempts affects results. A related topic is inference scaling, which is more relevant in agentic evaluation contexts. More detail on this is held in UK AISI's blogpost on inference scaling in cyber tasks.
- Model provider and serving configuration: The provider used to serve a model can affect both logistics (cost, throughput, data retention) and semantics (context lengths, quantisation). Different providers may serve the same underlying model with materially different capabilities; evaluators should document the provider used and any known serving details.
- Safeguards: Evaluators should document whether model safeguards are enabled or disabled, and justify this choice based on the evaluation objective. Where jailbreaking attempts have been made to bypass safeguards, this should also be documented, including the methods used and their observed effects.

Evaluators should also consider how tasks are presented to models and how outputs are scored. Poorly designed task or scoring settings can limit an evaluator's ability to accurately measure model capabilities. While more detail on this topic is held within NIST AI 800-2, evaluators should consider the following:

- Trials and epochs: run multiple trials per item to reduce uncertainty and quantify sampling variation. The number of trials should be balanced against evaluation budget, and the choice should be documented and tracked in the evaluation configuration. Evaluators should consider running a small trial first, along with automated checks or human spot checks, to limit costs. (for more detail, refer to Section 2.1.2 Singapore IMDA's "Starter Kit for Testing LLM-Based Applications for Safety and Reliability".)
- LLM-as-a-judge: when items lack programmatically verifiable answers, LLM judges with detailed rubrics can be used to score outputs. Evaluators should ensure that judge outputs are validated against human judgements on a representative sample. Evaluators may also consider using multiple judges or multiple rubrics, computing inter-rater agreement, and applying techniques such as prediction-powered inference to improve reliability. Evaluators should also

pay particular attention to use of non-deterministic auto-graders. Use of such graders should be weighed up to use of programmatic or rule-based auto-graders, particularly with regards to the complexity of the outputs being scored. (For more detail, refer to 2.1.4. Singapore IMDA's "Starter Kit for Testing LLM-Based Applications for Safety and Reliability".)

- Number of test items: Evaluators should attempt to balance statistical power against cost. Where possible, evaluators should use power analysis to determine an appropriate number of items.
- Aggregate statistics: Evaluators should select metrics aligned with the evaluation objective (e.g., mean accuracy for typical-case performance, or worst-case statistics for risk assessment).
- Item-level scoring and aggregation: Evaluators should note that the choice of per-item aggregation method (e.g., pass@k, majority-of-N, mean across trials) affects what is being measured. Evaluators should select aggregation methods that reflect the intended construct and should clearly document both the metric and the aggregation method used.

C.2. Write the evaluation code

When writing evaluation code – the code that implements the evaluation protocol – evaluators should adhere to the following established software engineering practices. More detail is held in NIST AI 800-2.

- Evaluators should ensure that different versions of evaluations are clearly labelled using a consistent versioning scheme, such as semantic versioning (e.g., Evaluation_v1.0.0, where changes to the major version indicate significant changes to evaluation design, minor versions indicate additions or adjustments, and patch versions indicate bug fixes). This ensures that results can always be traced back to the exact version of the evaluation that produced them.
- Evaluators should also design evaluations so that components are separated into distinct, interchangeable modules. Evaluation frameworks such as Inspect and Moonshot can support modular evaluation design.
- Finally, evaluators should include clear documentation alongside evaluation code, describing what the evaluation measures, how it is structured, and any known limitations. This supports both internal quality assurance and external review.

C.3. Run the evaluation and track results

Once the evaluation protocol has been designed and the evaluation code has been written, evaluators are ready to execute their evaluations. How evaluations are run - and how their outputs are captured and organised - has significant downstream implications for the reliability, reproducibility, and interpretability of results. Poorly organised runs can result in wasted compute, irreproducible findings, and conclusions that cannot be properly verified by third parties. This section sets out best practices for

the period immediately before and during evaluation execution, as well as practices that support robust interpretation of results.

In particular, evaluators should pay attention to the following areas when conducting their evaluations,

- Evaluation frameworks and software (e.g., Inspect or Moonshot) provide standardised infrastructure for querying models, handling errors, and logging results. Using such frameworks reduces the risk of bespoke implementation errors, supports reproducibility, and makes it easier for others to replicate or extend evaluation results.
- Summary statistics (e.g., mean accuracy) are useful for headline comparisons but are insufficient on their own. Full evaluation logs - including model inputs, outputs, and environment interactions - should be retained. These should include task logs (logs of the evaluated model performing the evaluation tasks) and judge logs (logs of the LLM-as-a-judge assessing the evaluated model's outputs). Alongside logs, evaluators should record the model version, provider, code commit hash, and all relevant protocol settings (e.g., inference settings, prompt templates, scoring configurations). Without these, logs may not be interpretable or reproducible in the future. More detail on best practice around log analysis is held within a joint paper from UK AISI and U.S. CAISI and Kirgis et al.
- Logs should be tagged with metadata that makes clear the purpose of each run, the settings used, and the relationship between different runs (e.g., whether logs belong to a debugging run, a development set run, or a final test run).
- Rather than launching a full evaluation run immediately, evaluators should begin with small runs on subsets of data to verify that the evaluation is functioning as intended. Incremental runs allow evaluators to catch bugs early, verify that logs are being captured correctly, and confirm that results are broadly in the expected range before committing significant compute. Evaluators should alternate between small debugging runs and larger validation runs until they are confident the evaluation is configured correctly, then proceed to the full run.

It should be noted that third-party evaluators may have resource constraints that necessitate prioritisation, it remains an open question on which items resource-constrained evaluators should prioritise.

C.4. Debug the evaluation

Even well-designed evaluations run into problems during execution. Bugs, misconfigurations, and unexpected model behaviours can all affect results, sometimes in ways that are not immediately visible in aggregate statistics. This section sets out the most common issues that evaluators are likely to encounter and provides practical guidance on how to identify and address them.

Debugging should be approached as an iterative activity that runs alongside evaluation execution rather than as a single step at the end. Evaluators should expect to uncover issues through a combination of manual transcript review, automated checks, and comparison to existing evidence (as discussed in Section C.3). When issues are identified, evaluators should document them, assess whether they warrant re-running affected portions of the evaluation, and record any mitigations applied.

The following are common issues that evaluators are likely to face.

- Results should be reported with appropriate uncertainty quantification, such as standard errors, confidence intervals, or credible intervals.
- Models may exploit implementation loopholes to score highly without genuinely solving the task. Examples include searching for benchmark answers online, disabling or removing failing test assertions, finding solution files left in the testing environment, or taking other unintended shortcuts. Mitigations include disabling internet access where not essential, sandboxing environments carefully, and using LLM-based transcript scanners to detect cheating at scale. More detail on this topic is held within a NIST paper.
- Model behaviour may shift when cues in the input suggest that the model is being evaluated. This is a challenge to external validity, as such cues are unlikely to be present in real-world use. The absence of verbalised awareness does not rule out the presence of evaluation awareness, and quantifying non-verbalised awareness remains an open line of research. Where verbalised evaluation awareness is observed, evaluators should report it alongside results and discuss its potential impact on validity.
- To search for these issues, evaluators should read transcripts - particularly transcripts of failed runs or anomalous outputs - to look for unexpected model behaviours. While most relevant for agentic evaluations, manual review of automated evaluations enables the identification of failure modes that may not be visible in aggregate metrics, such as models misinterpreting instructions, producing formatting errors, or exhibiting subtle refusal behaviours[4]. Failure modes should be categorised systematically so that patterns can be tracked across runs and addressed in subsequent iterations of the protocol. Again, more detail on best practice around this topic is discussed by a joint paper from UK AISI and U.S. CAISI.
    - At scale, manual review is not always feasible. Evaluators can supplement manual review with automated checks, including programmatic pattern detection (e.g. keyword matching for refusal phrases) and LLM-based scanners that assess transcripts for specific behaviours such as evaluation cheating or verbalised evaluation awareness. Where LLM-based scanners are used, their outputs should be validated against human labels on a

[4] While there is not robust method to avoiding unintended refusals, evaluators should look to conduct log analysis to mitigate potential issues around such model responses.

representative sample to ensure reliability. Multiple scanners or rubrics may be used in combination to improve accuracy.

- Models may be served in a degraded state due to quantisation choices, inference engine bugs, chat template misconfigurations, or provider-side configuration changes. This can alter model behaviour in ways that are difficult to detect from aggregate results alone. Degraded serving is a particular risk when evaluators self-host models, but it can also occur with commercial providers, who may vary in the context length, or other capabilities they offer for a given model. To mitigate unintentionally running evaluators on degraded models, evaluators should use the exploratory data set to ensure that results are broadly consistent with developer-reported scores or other published evaluations. Evaluators should then investigate substantial discrepancies.

Terminology

This section will cover terminology used in the best practice paper. Some terms that will be defined include,

- **AI agents:** a system that operates autonomously to achieve goals: it uses tools (external functions like web browsers or code executors), runs in a loop to make sequential decisions, and determines independently how to accomplish tasks. Agents are typically coordinated by scaffolds (framework managing agent calls, tools, and context) and deployed in sandbox environments for safe operation.
- **Automated evaluation:** Evaluations where scoring is performed without direct human judgement of correctness for individual test items.
- **Capability elicitation:** Experiments designed to unlock or enhance a model's ability after it has been trained, in order to best understand its capability profile. Capability elicitation does not always mean trying to assess the upper-bound of capabilities, only that the evaluator is trying to elicit capabilities of the model to a relevant level to the measurement construct.
- **Evaluation framework / software:** Software libraries for querying models, agent scaffolding and tools, error handling, and logging evaluation results instead of having to write those functions from scratch. Examples include Inspect and Moonshot.
- **Evaluation:** A set of tasks presented to a model. Capability evaluations are aimed at measuring its capabilities in a certain area. [5]
- **Inference scaling:** The return to evaluation performance from increased token budgets or reasoning effort settings.
- **Inference settings:** Settings that influence the process by which the model generates its outputs, e.g., temperature or reasoning effort.
- **LLM-as-a-judge:** the use of LLMs as graders of evaluations, rather than human graders. Sometimes referred to as "autograders".
- **Measurement construct:** An abstract concept not directly measurable (e.g., "mathematical reasoning").
- **Metadata:** Set of data in AI evaluations that gives information about data, including timestamps, token counts, model names and tasks.
- **Modularity:** The compatibility of an evaluation with multiple AI models and systems.
- **Multi-turn evaluations:** Evaluations which do not rely on a single static prompt, but use successive queries to mirror real-world situations (e.g., a conversation with a chatbot).
- **Multiple choice evaluations:** Question-answer evaluations which involve the model selecting between several potential answers.

[5] For simplicity, this document refers to 'evaluation' as a set of tests presented to a model. A 'testing run' refers to the set of evaluations done to a model, or set of models. It should be noted that others often use 'benchmark' to refer to a single evaluation, and 'evaluation' to refer to the overall testing activity.

- **Open-ended evaluations:** Evaluations which have many possible correct responses, typically when the evaluation domain cannot be divided into discrete tasks, and/or it is difficult to define objective grading criteria or verification procedures. Testing "rubrics" are often used to assess the validity of model responses.
- **Preregistration:** A widely adopted process across research fields, preregistration refers to researchers agreeing on the research approach and objectives prior to collecting data. In the context of AI evaluations, it is considered best practice. Concretely, it involves documenting, reviewing, iterating, and approving the experiments and draft reports of results before starting to onboard the model to be tested.
- **Question-answer pairs:** Evaluations which involve presenting the model with questions with set answers, often used to test knowledge retrieval.
- **Reasoning effort:** The amount of "reasoning" that a reasoning model uses when answering an evaluation prompt. The level of reasoning effort can typically be adjusted.
- **Refusals:** Refers to when LLMs refuse to answer or complete certain tasks due to safeguards.
- **Rubrics:** a structured grading framework developed with domain experts that defines a set of discrete scoring levels, where each level specifies explicit criteria a response must meet, used to evaluate the quality of an LLM's output-either by human graders or by an automated LLM-as-a-judge (autograder).
- **Safeguards**: Technical interventions implemented by AI developers to prevent users eliciting harmful information or actions from models.
- **Sandboxing:** When agents are able to run arbitrary code, sandboxing them in containers or virtual machines- an isolated environment that prevents them from affecting the rest of the system-reduces security risks and makes evaluations more portable.
- **Scoring settings:** settings that determine how test items are scored

- **Task settings:** Settings that determine how benchmark items are presented to models/systems and how models/systems can go about solving items.
- **Test items:** An individual question or problem in an evaluation, sometimes referred to as a sample or example. For example, LLM evaluations are often represented by a set of multiple-choice questions with one or more correct answers.
- **Testing run:** A set of evaluations presented to a model, aimed at measuring its capabilities across multiple areas.
- **Transcript/logs:** Refers to any records generated when using an AI system. Logs can include model inputs (user messages, prompts, instructions), model outputs (model responses, internal commentary, chain-of-thought reasoning, tool calls), environment interactions (tool outputs, API responses, terminal commands) and metadata (timestamps, token usage, error codes). Logs can also include task descriptions and scores from previous analyses (e.g., in agentic evaluations where agents are prompted to solve specific tasks and receive pass/fail scores).
- **Trials:** A single attempt at solving a test item. Evaluations often involve multiple attempts at each test item per AI system tested.

Bibliography

1. Bean, A.M., Kearns, R.O., Romanou, A., Hafner, F.S., Mayne, H., Batzner, J., Foroutan, N., Schmitz, C., Korgul, K., Batra, H., Deb, O., Beharry, E., Emde, C., Foster, T., Gausen, A., Grandury, M., Han, S., Hofmann, V., Ibrahim, L., Kim, H., Kirk, H.R., Lin, F., Liu, G.K.-M., Luettgau, L., Magomere, J., Rystrøm, J., Sotnikova, A., Yang, Y., Zhao, Y., Bibi, A., Bosselut, A., Clark, R., Cohan, A., Foerster, J., Gal, Y., Hale, S.A., Raji, I.D., Summerfield, C., Torr, P.H.S., Ududec, C., Rocher, L. and Mahdi, A. (2025) 'Measuring what Matters: Construct Validity in Large Language Model Benchmarks', in 39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks. arXiv:2511.04703. Available at: https://arxiv.org/abs/2511.04703.
2. Chouldechova, A., Atalla, C., Barocas, S., Cooper, A.F., Corvi, E., Dow, P.A., Garcia-Gathright, J., Pangakis, N., Reed, S., Sheng, E., Vann, D., Vogel, M., Washington, H. and Wallach, H. (2024) 'A Shared Standard for Valid Measurement of Generative AI Systems' Capabilities, Risks, and Impacts', *arXiv preprint*, arXiv:2412.01934. Available at: https://arxiv.org/abs/2412.01934.
3. Dubois, M., Coppock, H., Giulianelli, M., Flesch, T., Luettgau, L. and Ududec, C. (2025) 'Skewed Score: A Statistical Framework to Assess Autograders', *arXiv preprint*, arXiv:2507.03772. Available at: https://arxiv.org/abs/2507.03772.
4. Dubois, M., Zorer, E., Hamin, M., Skinner, J., Souly, A., Wynne, J., Coppock, H., Sato, L., Kapoor, S., Dev, S., Juchems, K., Mai, K., Flesch, T., Luettgau, L., Teague, C., Patey, E., Allaire, J.J., Pacchiardi, L., Hernandez-Orallo, J. and Ududec, C. (2026) 'Seven Simple Steps for Log Analysis in AI Systems', *TechRxiv*. doi: 10.36227/ https://arxiv.org/abs/2604.09563
5. European Commission (2025) General-Purpose AI Code of Practice – Safety and Security Chapter (Final version), 10 July. Available at: https://digital-strategy.ec.europa.eu/en/policies/contents-code-gpai (Accessed: 27 March 2026).
6. Gekker, G., Segal, M., Lahav, D. and Nevo, O. (2025) 'What Makes an Evaluation Useful? Common Pitfalls and Best Practices', *arXiv preprint*, arXiv:2503.23424. Available at: https://arxiv.org/abs/2503.23424.
7. Guerdan, L., Barocas, S., Holstein, K., Wallach, H., Wu, Z.S. and Chouldechova, A. (2025) 'Validating LLM-as-a-Judge Systems under Rating Indeterminacy', arXiv preprint, arXiv:2503.05965. Available at: https://arxiv.org/abs/2503
8. Hamin, M. and Edelman, B. (2025) 'Cheating on AI Agent Evaluations', National Institute of Standards and Technology, Center for AI Standards and Innovation. Available at: https://www.nist.gov/caisi/cheating-ai-agent-evaluations (Accessed: 27 March 2026).
9. Infocomm Media Development Authority (IMDA) (2026) Model AI Governance Framework for Agentic AI. Version 1.5, 20 May. Available at: https://go.gov.sg/mgfagentic-feedback (Accessed: 27 March 2026).

10. Infocomm Media Development Authority (IMDA) (2026) Starter Kit for Testing LLM-Based Applications for Safety and Reliability. Version 1.0. AI Verify Foundation and IMDA. Available at: https://www.aiverifyfoundation.sg (Accessed: 27 March 2026).
11. Kapoor, S., Kirgis, P., Schwartz, A., Rabanser, S., Allaire, J.J., Bommasani, R., Dubois, M., Hadfield, G., Hall, A., Hooker, S., Lazar, S., Newman, S., Papailiopoulos, D., Tekofsky, S., Toner, H., Ududec, C. and Narayanan, A. (2026) 'Open-world evaluations for measuring frontier AI capabilities', *CRUX*. Available at: https://cruxevals.com/open-world-evaluations.pdf (Accessed: 27 March 2026).
12. Kirgis, P., Kapoor, S., Rabanser, S., Nadgir, N., Ududec, C., Dubois, M., Allaire, J.J., Stosz, C., Hobbhahn, M., Steinhardt, J. and Narayanan, A. (2026) 'Log analysis is necessary for credible evaluation of AI agents', arXiv preprint, arXiv:2605.08545. Available at: https://arxiv.org/abs/2605.08545.
13. Luettgau, L., Coppock, H., Dubois, M., Summerfield, C. and Ududec, C. (2025) 'HiBayES: A Hierarchical Bayesian Modeling Framework for AI Evaluation Statistics', *arXiv preprint*, arXiv:2505.05602. Available at: https://arxiv.org/abs/2505.05602.
14. METR (2024) *Example Protocol for AI Agent Evaluations*. Model Evaluation and Threat Research. Available at: https://metr.github.io/autonomy-evals-guide/example-protocol/ (Accessed: 27 March 2026).
15. METR (2025) *Measuring Autonomous AI Capabilities: Best Practices for Running AI Agent Evaluations*. Model Evaluation and Threat Research. Available at: https://metr.org/measuring-autonomous-ai-capabilities/#best-practices-for-running-ai-agent-evaluations (Accessed: 27 March 2026).
16. Mougan, C., Morlock, L., Aguirre, J., Black, J.R.M., Brauner, J., Campos, S., Dev, S., Fernández Llorca, D., Franzin, A. … Schellaert, W. (2026) 'The science and practice of proportionality in AI risk evaluations', Science, 391(6787), pp. 769–771. https://doi.org/10.1126/science.aea3835.
17. NIST (2025) 'Analyzing Transcripts of AI Agent Evaluations', *CAISI Research Blog*. National Institute of Standards and Technology. Available at: https://www.nist.gov/blogs/caisi-research-blog/analyzing-transcripts-ai-agent-evaluations (Accessed: 27 March 2026).
18. NIST (2026) *Practices for Automated Benchmark Evaluations of Language Models*. (U.S. Department of Commerce, Washington, D.C.), NIST AI 800-2, Initial Public Draft. https://doi.org/10.6028/NIST.AI.800-2.ipd.
19. Paskov, P., Byun, M., Wei, K. and Webster, T. (2025) *Preliminary suggestions for rigorous GPAI model evaluations*. RAND Corporation, PE-3971-1. Available at: https://www.rand.org/pubs/perspectives/PEA3971-1.html (Accessed: 27 March 2026).
20. Pencharz, J., Flesch, T. and Sandbrink, J. (2024) 'Long-Form Tasks: A Methodology for Evaluating Scientific Assistants', AISI Blog. AI Security Institute.

Available at: https://www.aisi.gov.uk/blog/a-methodology-for-evaluating-scientific-assistants (Accessed: 27 March 2026).

21. Rabanser, S., Kapoor, S., Kirgis, P., Liu, K., Utpala, S. and Narayanan, A. (2026) 'Towards a Science of AI Agent Reliability', *arXiv preprint*, arXiv:2602.16666. Available at: https://arxiv.org/abs/2602.16666.
22. Salaudeen, O., Reuel, A., Ahmed, A., Bedi, S., Robertson, Z., Sundar, S., Domingue, B., Wang, A. and Koyejo, S. (2025) 'Measurement to Meaning: A Validity-Centered Framework for AI Evaluation', arXiv preprint, arXiv:2505.10573. Available at: https://arxiv.org/abs/2505.10573.
23. UK AI Safety Institute (2025) 'Evidence for Inference Scaling in AI Cyber Tasks: Increased Evaluation Budgets Reveal Higher Success Rates', *AISI Blog*. Available at: https://www.aisi.gov.uk/blog/evidence-for-inference-scaling-in-ai-cyber-tasks-increased-evaluation-budgets-reveal-higher-success-rates (Accessed: 27 March 2026).
24. UK AI Safety Institute (2025) 'Our Approach to AI Capability Elicitation', *AISI Blog*. Available at: https://www.aisi.gov.uk/blog/our-approach-to-ai-capability-elicitation (Accessed: 27 March 2026).
25. Wallach, H., Desai, M., Cooper, A.F., Wang, A., Atalla, C., Barocas, S., Blodgett, S.L., Chouldechova, A., Corvi, E., Dow, P.A., Garcia-Gathright, J., Olteanu, A., Pangakis, N., Reed, S., Sheng, E., Vann, D., Vaughan, J.W., Vogel, M., Washington, H. and Jacobs, A.Z. (2025) 'Position: Evaluating Generative AI Systems Is a Social Science Measurement Challenge', in *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. arXiv:2502.00561. Available at: https://arxiv.org/abs/2502.00561.
26. Weidinger, L., Raji, I.D., Wallach, H., Mitchell, M., Wang, A., Salaudeen, O., Bommasani, R., Ganguli, D., Koyejo, S. and Isaac, W. (2025) 'Toward an Evaluation Science for Generative AI Systems', arXiv preprint, arXiv:2503.05336. Available at: https://arxiv.org/abs/2503.05336.
27. Zhu, Y., Jin, T., Pruksachatkun, Y., Zhang, A., Liu, S., Cui, S., Kapoor, S., Longpre, S., Meng, K., Weiss, R., Barez, F., Gupta, R., Dhamala, J., Merizian, J., Giulianelli, M., Coppock, H., Ududec, C., Sekhon, J., Steinhardt, J., Kellermann, A., Schwettmann, S., Zaharia, M., Stoica, I., Liang, P. and Kang, D. (2025) 'Establishing Best Practices for Building Rigorous Agentic Benchmarks', *arXiv preprint*, arXiv:2507.02825. Available at: https://arxiv.org/abs/2507.02825.