AI STUPID LEVEL

PUBLIC BENCHMARK METHODOLOGY / 2026

Measuring change. Informing decisions.

AI Stupid Level public methodology for continuous model evaluation

From executable evidence to longitudinal model intelligence

Controlled benchmarking and explicit data provenance.

Version-aware drift analysis and failure attribution.

Production routing informed by measured quality and delivery outcomes.

METHODS / MEASUREMENTS / PUBLIC REVIEW

Public methodology edition - 2026

Prepared for public technical review, research discussion and community critique

Live task identities, hidden tests and selected production calibration details are intentionally withheld.

PUBLIC METHODOLOGY

1. Executive technical overview

AI Stupid Level (ASL) connects repeated model evaluation, longitudinal change detection and request-level model selection in one operating platform. The core question is not only which model scores highest once, but whether a model continues to perform a known workload over time, how repeatable its outcomes are, and how that evidence should influence model selection.

What this public edition is for

This document explains the measurement principles, benchmark surfaces, statistical interpretation and evidence-to-routing logic needed for external review.

It intentionally does not publish the complete live task bank, hidden test cases, full prompt-variation catalogue, exact detector calibration constants, internal source paths, operational schedules or production routing constants. The objective is to make the methodology inspectable without turning the live benchmark into a training target or a replication cookbook.

An integrated measurement cycle. Controlled requests produce task-level execution evidence. Repeated trials expose variability. Configuration identities define comparable time series. Longitudinal analysis asks whether behavior changed within the same instrument. Attributed production outcomes can then refine routing decisions.

Recorded evidence in 2026	Count	Unit
Canary score records	57,067	Lightweight early-warning results
Logged tool executions	93,088	Individual tool calls with result and latency
Captured raw outputs	45,378	Returned model text and extraction outcomes
Per-test-case results	30,039	Individual pass/fail records

These are distinct evidence types and are not added together as a single benchmark total. Every empirical claim in this document should be read with its own cohort, sampling unit and observation window.

FROM A CONTROLLED TASK TO AN EXPLAINABLE MODEL CHOICE

2. Measurement framework

Layer	Core question
Request contract	Were models evaluated under declared and comparable conditions?
Capability	Did the returned work satisfy the task and evaluator?
Repeatability	Were outcomes, tool choices and resource use consistent?
Scoring and uncertainty	What is aggregated, and what does the uncertainty represent?
Longitudinal analysis	Did behavior change within a comparable instrument?
Reporting and routing	How are measured signals interpreted and used for selection?

Four principal benchmark surfaces

Surface	Configured cadence	Unit of work
Canary	Hourly	Small early-warning probe, one trial per task
Coding	Every few hours	10 tasks x 5 trials per run
Reasoning	Daily	Four multi-turn session types
Tooling	Daily	Tiered tasks with a real tool surface in an isolated sandbox

Interpret each layer separately

Execution-based correctness, deterministic heuristics, repeatability metrics, statistical detector alerts and routing scores answer different questions. A provider fingerprint supports attribution; an uncertainty interval describes uncertainty under assumptions; a routing score expresses a configured decision rule.

A scientific result needs a declared unit of observation, a comparable evaluation contract and a traceable output. ASL preserves model identity, task identity, request conditions, execution evidence, measurement outputs, methodology identity and time so that a score can be connected back to the evidence that produced it.

THE BENCHMARK IS AN INSTRUMENT

3. Fairness, integrity and contamination controls

Cross-model comparison requires a declared request contract and explicit treatment of provider exceptions. ASL applies a common request policy where provider support allows it, while recording cases where a provider or reasoning mode rejects or overrides a requested setting.

Control	Public description	Why it matters
Common request contract	Temperature, output budget and allowed request fields are controlled where supported.	Reduces avoidable cross-model configuration differences.
Provider exceptions	Unsupported or provider-imposed behaviors are identified rather than silently normalized.	Prevents false comparability.
Batch-level variation	Surface identifiers and prompt envelopes are deterministically varied between batches.	Reduces exact-response reuse and dependence on one phrasing.
Prompt equality within a batch	Every model receives the same task variant for the same batch.	Keeps the comparison fair within that run.
Execution-based evaluation	Generated code is executed against test cases where direct execution is available.	Avoids making a changing LLM judge the primary correctness authority.
Sandboxing	Tool and code execution are network-isolated and resource-bounded.	Limits side effects and keeps evaluation conditions controlled.
Methodology identity	Score-moving benchmark changes create a new configuration identity.	Prevents methodology edits from being misread as model drift.

What these controls do not prove

No single control establishes immunity to benchmark contamination, provider-side recognition, hidden serving changes or task-family memorization. The goal is to reduce these risks, make them observable where possible and avoid overstating what the instrument can prove.

ASL also records served-model metadata or provider fingerprints when available. A changed identifier is useful evidence for investigating a serving transition, but it does not by itself prove that weights changed or that a measured quality shift was caused by that transition.

DIFFERENT TASKS ANSWER DIFFERENT QUESTIONS

4. Capability surfaces

Coding

The primary coding suite evaluates Python problem solving through executable tests. Correctness is the dominant signal because it is directly observable. Secondary dimensions capture properties such as structure, task understanding, efficiency, edge-case handling, debugging behavior, format compliance and safety-related behavior. Exact live task identities, hidden tests and weighting constants are not published in this edition.

Reasoning

The reasoning suite uses multi-turn sessions to test sustained behavior rather than a single isolated answer. The sessions examine specification adherence, use of information introduced earlier in the exchange, coherent code modification and long-context task completion. Where possible, code turns are validated by execution; other turn types use declared deterministic checks or heuristics.

Tooling

The tooling suite gives a model a real, sandboxed tool surface spanning file inspection, modification, search and command execution. It measures whether the overall task worked, whether appropriate tools were selected, whether parameters were correct, how efficiently the model acted, whether it recovered from errors, and whether it respected context and safety constraints.

Canary

The canary is a lightweight hourly probe designed for early warning, not for a confident degradation conclusion by itself. It is intentionally small and statistically weak. Its role is to surface a candidate change quickly so that the larger repeated suites and longitudinal record can provide stronger evidence.

Capability is not availability

A provider delivery failure, caller authentication problem, quota issue or local transport problem is not automatically treated as evidence that the model lacks capability.

ASL separates valid task outcomes from unavailable measurements and carries that distinction into reliability analysis and routing.

CONSISTENCY IS ITS OWN DIMENSION

5. Reliability and repeatability

Capability asks whether the work was completed. Repeatability asks whether comparable runs produce stable outcomes, similar action patterns and similar resource requirements. ASL keeps these dimensions separate rather than compressing them into a single reliability number.

Dimension	Method family	Question
Outcome consistency	Agreement across repeated success/failure outcomes	Does the model reach the same result on comparable attempts?
Trajectory distribution	Pairwise Jensen-Shannon similarity over tool-use distributions	Does it use similar mixtures of tools?
Trajectory sequence	Normalized edit-distance similarity over ordered actions	Does it use tools in a similar order?
Resource consistency	Variation across latency, token use and action count	Is the resource cost of completing the work stable?

Failure direction matters. A model can be highly consistent because it succeeds repeatedly or because it fails repeatedly. ASL therefore interprets outcome agreement alongside pass rate, task coverage and the direction of the result.

Coverage guards matter. Model-level consistency should not be inferred from a tiny surviving subset of tasks. Results are interpreted with task coverage and nullable values where the available comparison set is insufficient.

Disagreement between dimensions is useful. A model can choose similar tools but in a changing order, or maintain similar outcomes while using highly variable resources. Those differences are information, not noise to be hidden by an all-purpose reliability score.

No extra model calls are required for retrospective repeatability analysis

The reliability layer can be computed from already-recorded trial and tool-execution evidence. This keeps the repeatability view tied to the same observed work rather than introducing a separate evaluation surface.

A SCORE IS A CONFIGURED SUMMARY

6. Score composition and uncertainty

ASL publishes suite-specific views and a combined quality view. Coding, reasoning and tooling remain distinct measurements; the combined view gives coding the greatest influence and uses reasoning and tooling as additional capability surfaces. Exact production weighting constants are intentionally omitted from the public edition.

View	Interpretation
Coding	Executable coding performance under the current coding instrument
Reasoning	Performance over defined multi-turn reasoning sessions
Tooling	Task completion and tool-use behavior in the sandboxed tool environment
Combined	Configured summary used for broad quality ranking; not a universal measure of intelligence

Uncertainty metadata

Score records can carry nominal t-based confidence bounds, standard error, sample size and historical variance. These values must be interpreted with the estimator and sampling unit that produced them.

Repeated trials on the same task, related tasks and serially dependent daily observations are not automatically independent samples. A familiar mean-based interval can be a useful reference, but a daily median or transformed composite requires uncertainty appropriate to that estimator. ASL therefore treats score, interval, sample count, suite and benchmark configuration as a bundle rather than reading an interval in isolation.

Interpretation rule

A composite score is a configured decision summary. A repeatability score is not a pass rate. A detector statistic is not causal attribution. Each quantity should be used for the question it was designed to answer.

DRIFT ONLY MAKES SENSE WITHIN A STABLE INSTRUMENT

7. Versioning and baseline continuity

A change detector asks whether a model moved relative to its own past. That comparison is meaningful only while the measurement instrument is sufficiently stable. If a task changes, a test becomes stricter, a prompt contract changes or a scoring rule is retuned, every model can move even when the models themselves did not.

Configuration identity

ASL assigns a content-based benchmark configuration identity to score-moving inputs. When a covered input changes, the affected time series begins a new baseline. Non-score-moving edits such as comments, logging changes or unrelated refactors should not reset historical continuity.

Scenario	Expected interpretation
Material score drop with the same configuration	Eligible for longitudinal change detection
Identical score drop at a configuration change	Baseline transition; do not call it model drift without separate evidence
Historical row with unknown configuration identity	May remain visible, but comparability is weaker and must be stated

Why this matters publicly

A leaderboard can hide instrument changes inside a single number. Version-aware analysis makes the comparison boundary explicit, so a methodology revision does not silently rewrite the model history.

Within-model drift analysis therefore requires more than a timestamp. It requires the model, suite, effective request policy, benchmark configuration and relevant provider metadata needed to establish that the observations belong to the same measurement regime.

CHANGE DETECTION, NOT CAUSAL STORYTELLING

8. Drift detection

ASL uses a downward Page-Hinkley style detector over a sequence of daily medians. The detector is designed to accumulate evidence when a model remains below its learned historical level and to reset after a confirmed change so that it can learn the new regime.

Calibration and regimes

Detector sensitivity is calibrated against historical behavior. Model-specific historical variability is used when interpreting stable, volatile, degraded and recovering regimes. Exact production thresholds, cold-start lengths, rearm settings and secondary-scan constants are withheld from the public edition so that the live detector remains less gameable.

A second screening layer

A more sensitive multi-scale scan is used as a screening mechanism. It is intentionally allowed to over-fire. Candidate capability changes require both statistical evidence and a material effect size; significance alone is not treated as sufficient evidence of a meaningful degradation.

Historical validation cohort

Quantity	Observed result
Proxy-evaluable window	73 elapsed days in 2026
Observed serving-version transitions	4
Version-linked degradation events	0
Uncorroborated detector alerts	7
Armed exposure	1,332 model-days
Uncorroborated alert density	0.526%, approximately 1 per 190 model-days

This is an alert-density observation, not a validated false-positive rate. An uncorroborated event is not automatically false, and the absence of observed positive degradation events means recall cannot be estimated from this cohort.

WHAT REPEATED MEASUREMENT REVEALS

9. Longitudinal observations

Within-day and between-day variation

A historical analysis covered 31,352 hourly score rows across 49 models. The reported standard deviation was 2.80 points within a day and 8.43 points between daily medians, a descriptive ratio of 3.01.

The result suggests a day-level component worth investigating, but it does not establish a formal variance-components model or identify the cause. Task mix, missingness, serial dependence, provider behavior and methodology changes can all affect the comparison.

Study	Observation	Interpretation boundary
Within-day vs between-day	2.80 vs 8.43 points; ratio 3.01	Descriptive evidence of temporal structure, not causal attribution
Cache-collapse check	805 batches examined; output length varied across trials in 78.1% of batches	Rules out universal identical-completion reuse in those batches; does not rule out all caching or benchmark recognition
Correctness agreement	16 models with both per-trial records and axis scores agreed within roughly 0.6-0.8 points	Supports approximate agreement in that observed subset, not universal interchangeability

Why continuous observation matters

Historical model series can spend most days inside a narrow band while still producing short-lived low-score episodes. Weekly or monthly snapshots can miss those events entirely. Continuous measurement is therefore valuable even when a model returns to its prior level before the next conventional benchmark refresh.

Do not over-read a single dip

A low task result can be localized to one workload, one provider condition or one temporary boundary effect. Aggregate change should be traced back to task composition, configuration identity and delivery metadata before being generalized.

SEPARATE MODEL BEHAVIOR FROM DELIVERY CONDITIONS

10. Attribution and reporting

ASL combines per-request failure classification with cross-model correlation. The purpose is to distinguish a valid task failure from a provider delivery problem, a caller configuration error or a shared infrastructure condition.

Class	Representative cause	Treatment
Model/provider delivery	Provider error, overload, timeout or throttling	Eligible for delivery reliability analysis when attribution is sufficient
Caller request	Malformed payload or unsupported request construction	Do not treat as model quality
Caller authentication	Missing or invalid caller credentials	Do not penalize the model
Caller quota	Caller account quota or entitlement	Do not penalize the model
Transport	Local network or transport fault	Do not penalize the model
Unknown	Insufficient evidence	Leave unassigned rather than forcing attribution

Provider-wide interpretation requires correlation across multiple tracked models rather than a single failing endpoint. Simultaneous movement across providers motivates a broader platform investigation. Correlation helps localize a problem, but it does not establish cause.

Reporting rules

Rule	Meaning
Compare with the model's own history	A baseline represents that model's measured level under the comparable instrument
Match the suite	Coding, reasoning and tooling distributions answer different questions
Match the configuration	Methodology changes define comparison boundaries
Declare the basis	State whether the result is a threshold, trend, variance or availability statement
Preserve coverage and time	Report the sampling unit, sample count, observation window and chronology

MEASUREMENT BECOMES A DECISION INPUT

11. The Smart Router

ASL exposes an OpenAI-compatible routing layer that can select a model per request. The public methodology describes the decision structure while withholding exact production calibration constants and ranking formulas.

Decision input	Role in selection
Benchmark quality	Ranks eligible models on the capability surface relevant to the caller's strategy
Caller constraints	Filters by provider availability, cost, latency, tool support, streaming and exclusions
Fallback policy	Allows an eligible alternative when the primary request fails
Observed delivery outcomes	Can adjust rankings when sufficient model-attributable production evidence exists
Request-level logging	Preserves selected model, rationale, candidate pool, latency, tokens, estimated cost and outcome metadata

Evidence earns influence

Sparse production feedback is shrunk strongly toward the benchmark prior, so one or two requests cannot overturn a mature benchmark ordering. As model-attributable observations accumulate, delivery reliability and observed latency can receive more influence. Caller ratings are bounded because they are sparse, self-selected and potentially gameable.

Why the benchmark and router remain conceptually separate

The benchmark measures model behavior under controlled conditions. The router makes a production decision under a caller's actual constraints. A model can have the highest benchmark quality and still be the wrong choice for a specific request because of latency, cost, unavailable credentials, missing tool support or recent delivery reliability.

The routing product is a heuristic, not a probability

Router performance should be evaluated on delivered outcomes for the target workload. The decision logic is inspectable, but the final ranking is not presented as a calibrated probability that a request will succeed.

WHAT ASL CLAIMS - AND WHAT IT DOES NOT

12. Scientific boundaries and public reproducibility

Surface	Current interpretation boundary
Coding	Python tasks and their implemented tests; no cross-language generalization claim
Reasoning	Defined multi-turn tasks with execution-based or deterministic heuristic evaluators
Tooling	The exposed sandboxed tool surface and tiered task families
Repeatability	Outcome, trajectory and resource consistency inside comparable windows
Safety	Limited benchmark axes plus separate adversarial protocols; not a comprehensive safety guarantee
Cost	Maintained price estimates; not a direct measurement of every provider invoice or discount
Creative quality	No dedicated creative-writing benchmark in this edition

Public reproducibility contract

A reproducible public result should identify the workload family, model and provider identity, request contract, evaluator type, analysis cohort, benchmark configuration, estimator or detector family, observation window and sampling unit. Historical claims should retain enough evidence to distinguish a definition, a scheduled attempt and an observed empirical result.

Intentionally withheld from the public edition

Complete live task identities, hidden tests, full prompt variants, exact detector calibration constants, internal source maps, table names, cron schedules and production routing constants. These details can be shared selectively for technical diligence or research review when appropriate, while keeping the public benchmark less vulnerable to benchmark-specific optimization.

References

[R1] Rabanser et al. (2026), Towards a Science of AI Agent Reliability. Multidimensional reliability ideas informing ASL's consistency and robustness adaptations.

[R2] NIST/SEMATECH e-Handbook, CUSUM Average Run Length. Statistical process-monitoring reference for interpreting observations before a signal.

[R3] SciPy documentation, scipy.stats.sem. Reference for standard error and the distinction between mean-based uncertainty and uncertainty for other estimators.

Public methodology principle

ASL aims to publish enough methodology for serious external criticism while preserving the integrity of the live measurement instrument. The benchmark should be inspectable, but it should not become easier to optimize against than the real workloads it is intended to measure.