Statistical Companion Document

Formal Statistical Foundations for the Mavai® Methodology

Version 1.7.1 · last updated 2026-10-22

Copyright © 2026, Michael Franz Mannion BSc (Hons) MBA
Mavai® is a registered trademark of Michael Mannion in Switzerland (No. 856092).

Contents

Document History . . . 3
Forward Scope . . . 6
Introduction: The Assumption of Certainty in Software Testing . . . 6
Two Dimensions of Stochasticity . . . 7
What This Document Contains . . . 7
The mavai Project Family . . . 8
Document Purpose and Audience . . . 8
The Model in Brief: Contracts, Tests, Experiments . . . 9
Clause Forms: Rate-Bounded and Categorical . . . 10
Two Testing Paradigms: Compliance and Regression . . . 11
1. Statistical Model . . . 13
1.1 Bernoulli Trial Framework . . . 13
1.2 Binomial Aggregation . . . 13
1.3 Assumptions and Limitations . . . 14
1.4 Criterion Decomposition of the Functional Dimension . . . 17
1.5 Baselines . . . 39
2. Baseline Estimation (Experiment Phase) . . . 44
2.1 Point Estimation . . . 44
2.2 Standard Error . . . 44
2.3 Confidence Intervals . . . 44
2.4 Sample Size Determination . . . 47
2.5 Two Binomial Counts Given Their Total . . . 48
3. Threshold Derivation for Regression Testing . . . 48
3.1 The Problem . . . 48
3.2 One-Sided Hypothesis Testing Framework . . . 49
3.3 Normal-Approximation Intuition for One-Sided Bounds . . . 50
3.4 Regression Cutoff (One-Sided Fisher Exact Test) . . . 50
3.5 Reference Table: Regression Cutoffs . . . 53
3.6 Testing Against a Given Threshold (Compliance) . . . 54
4. The Perfect Baseline Problem ($\hat{p} = 1$) . . . 56
4.1 Problem Statement . . . 56
4.2 Interpretation of 100% Observed Success . . . 57

4.3 The Wilson Lower Bound Solution (withdrawn) . . . 57
4.4 Extended Example: Highly Reliable API . . . 59
4.5 Theoretical Note: Beta-Binomial Alternative . . . 59
5. Test Execution and Interpretation . . . 60
5.1 Decision Rule . . . 60
5.2 Type I and Type II Errors . . . 61
5.3 Statistical Power . . . 61
5.4 Sample Size for Desired Power . . . 62
5.4.1 Self-Consistent Power for Baseline-Derived Thresholds . . . 62
5.5 Sample Size for SLA Verification . . . 65
5.6 The Design Alternative Rate . . . 65
5.7 Test Intent: VERIFICATION vs SMOKE . . . 66
6. The Three Operational Approaches: Mathematical Formulation . . . 71
6.1 Approach 1: Sample-Size-First (Cost-Driven) . . . 71
6.2 Approach 2: Confidence-First (Risk-Driven) . . . 72
6.3 Approach 3: Threshold-First (Externally-Dictated) . . . 72
7. Reporting and Interpretation . . . 73
7.1 Transparent Statistics Output . . . 73
7.2 Confidence Statement . . . 75
7.3 Multiple Testing Considerations . . . 76
7.4 Threshold Provenance . . . 78
8. Assumptions and Validity Conditions . . . 79
8.1 When Normal Approximation is Valid . . . 79
8.2 Independence Violations . . . 80
8.3 Non-Stationarity . . . 83
8.4 Guardrails for Assumption Validity . . . 84
8.5 Repeated Use and Sequential Monitoring . . . 91
9. Summary of Key Formulas . . . 93
The decision rules . . . 93
Estimation (MLE / plug-in estimation) . . . 94
Regression Cutoff (Exact conditional test — *regression/fisher*) . . . 94
Regression Power (Exact) . . . 94
Compliance Decision (Exact — *compliance/exact-binomial*) . . . 95
SMOKE Compliance Decision (Exact — *compliance/smoke-exact-binomial*) . . . 95
Compliance Feasibility (Exact) . . . 95
Compliance Power and Sizing (Exact) . . . 95
Wald Confidence Interval (two-sided) (Asymptotic — pedagogical, not used by the methodology) . . . 95
Wilson Score Interval (Wilson score construction — default descriptive interval) . . . 95
Rule of Three, for zero failures (Heuristic — quick approximation at 95% confidence) . . . 96
Sample Size for Precision (Asymptotic — planning approximation based on normal asymptotics) . . . 96
Empirical Percentile (nearest-rank) (Exact — definition) . . . 96
Latency Threshold (Non-parametric / distribution-free — *latency/precedence*. Exact for continuous F_T; conservative under ties.) . . . 96

Explicit Latency Requirement (Exact — *latency/compliance-exact-binomial*) 96
10. Transparent Statistics Mode 96
10.1 Purpose 96
10.2 Output Structure 97
10.3 Example output: the consult-advice tests 101
10.4 Mathematical Notation 107
10.5 Validation by Statisticians 107
10.6 Calibration Conformance 107
10.7 Conformance Checklist 109
11. Statistical Discipline Through Design 110
12. Latency: Empirical Percentile Analysis 111
12.1 The Statistical Challenge of Latency 112
12.2 Empirical Percentile Estimation 112
12.3 Latency Assertions 114
12.4 Threshold Derivation from Baselines 117
12.5 Sample Size Requirements for Percentile Estimation 120
12.6 Enforcement Modes 124
12.7 Relationship to Pass-Rate Testing 126
12.8 Numerical Conventions 127
Appendix A. Elements of the statistical model 128
References 133

Document History

#	Date	Milestone
1	**2025-12**	**First issue.** Pass rates modelled as Bernoulli trials and binomial counts, with Wilson intervals.
2	**2026-02**	**Latency added.** A second dimension, latency, measured by empirical percentiles.
3	**2026-04**	**Stricter latency.** Latency measured on successful calls only, with an exact order-statistic bound in place of the earlier approximation.
4	**2026-05**	**Worked examples corrected.** The examples in §§4.3–4.4 now use the same Wilson construction as the rest of the document.
5	**2026-05**	**Independence assumption explained.** New §1.3.1 sets out when treating trials as independent is reasonable.
6	**2026-05**	**Several criteria per contract.** Each criterion has its own pass rate and a PASS, FAIL or INCONCLUSIVE verdict; compliance and regression become separate procedures.
7	**2026-05**	**Multi-criterion clarifications.** Every trial counts towards each criterion, one sampling is shared by all criteria, and INCONCLUSIVE means only that no verdict could be reached.
8	**2026-05**	**Criterion scope withdrawn.** Every sample is seen by every criterion.
9	**2026-05**	**Terminology.** ”Empirical” clauses are renamed ”rate-bounded”; new §1.4.5b covers guardrails.

#	Date	Milestone
10	**2026-05**	**Guardrail clarifications.** §1.4.5b now says which sampling measures what, and where a guardrail's rates are estimated.
11	**2026-05**	**Accumulation withdrawn.** Baselines are no longer accumulated over time; §1.5.5 covers expiry only.
12	**2026-06**	**Editorial consolidation.** Repeated formulas and examples replaced by cross-references.
13	**2026-07**	**Power for baseline-derived thresholds.** New §5.4.1 sizes a test against the threshold it actually applies.
14	**2026-07**	**Operational approaches reconciled.** §6 names its three approaches consistently.
15	**2026-08**	**Zero baseline stated.** New §4.3.4: a baseline with no successes cannot support a test, and the design is refused. This milestone appeared while the title still read 1.4.1.
16	**2026-09**	**1.5.0: new decision rules.** A statistical review showed that the 1.4.1 rules did not keep false alarms at the stated level, and recommended replacing Wilson with more statistically honest rules, each chosen for its circumstance: Fisher's test for regression, the exact binomial test for compliance and for explicit latency limits, and a rank rule for latency regression. Wilson remains the descriptive interval only, and decides nothing. Verdicts can differ from 1.4.1; earlier results can be reproduced from the mavai-R 0.10.13 fixtures.
17	**2026-10**	**1.6.0: every assertion enforced by default.** Correctness and latency assertions, given or baseline-derived, now count towards the verdict by default; in 1.5.0 latency assertions only warned unless enforcement was switched on. A run-time switch, not a per-test setting, can make correctness, latency or both advisory: an advisory assertion is still decided by its rule and reported, but it cannot fail the test. No decision rule changes; verdicts can, since a latency breach that warned under 1.5.0 now fails the test by default.
18	**2026-10**	**1.7.0: SMOKE decides by its own rule.** Under 1.5.0 a normative SMOKE test was decided like a VERIFICATION one, so a test too small to demonstrate its requirement could only FAIL: the 1.5.0 review rightly found that such a FAIL carries no evidence, and the companion said so instead of giving SMOKE a rule of its own. Under SMOKE the burden of proof is now reversed (§5.7.3): a normative criterion FAILs only on evidence that the service falls short of its requirement, and a PASS means only that no such evidence was found. Two rules are added, `compliance/smoke-exact-binomial` and `latency/smoke-compliance-exact-binomial`, both exact binomial tests and so existing machinery; VERIFICATION is unchanged.

#	Date	Milestone
19	**2026-10**	**1.7.1: SMOKE decisions enter neither envelope.** A clarification, not a new rule: a criterion decided by a SMOKE rule adds nothing to either Type-I envelope of §1.4.6, because its α bounds a false FAIL and its PASS claims no compliance. No verdict, count or decision rule changes.
20	**2026-10**	**Denominator-policy residue removed.** §8.4.5 no longer lists a denominator-policy mismatch among the hard invalidators: there has been no per-criterion denominator policy since 1.3.1 (§1.4.5a). No verdict, count or rule changes. This milestone appeared while the title still read 1.7.1.
21	**2026-11**	**Architectural commitments placed in Distributional Contracts.** The promised chapter on architectural commitments is not written here: how a guardrail commitment is declared and what its record holds are contractual matters, now set out in the composition chapter of Distributional Contracts (§8.9), and the three pointers to the chapter say so. §1.4.5, §1.4.5b and §1.4.8 now agree with Clause Forms that a guardrail's own rate-bounded contract leads and an observational zero-failures criterion complements it. No verdict, count or decision rule changes. This milestone appeared while the title still read 1.7.1.
22	**2026-11**	**A categorical clause can sit on any contract.** Earlier text said categorical clauses are discharged architecturally and routed out of the service's own experiment, as though they could not be stated on a service's contract. Like any clause, a categorical clause can be stated on any contract and is evaluated observationally wherever it is stated. A guardrail is a separate pattern, used when the service cannot realistically be made to produce the prohibited content but a guardrail can be tested with it directly; its statistics stay here, the pattern itself is Distributional Contracts'. The one-line statement of the methodology is revised. No verdict, count or decision rule changes. This milestone appeared while the title still read 1.7.1.
23	**2026-11**	**The guardrail's two error rates.** A paragraph of §1.4.5b said which criteria a safety-class contract requires; that is contractual and now sits in Distributional Contracts §8.9, which ties the allocation of a guardrail's two error rates to the Neyman–Pearson principle. §1.4.5b states the statistical consequence instead: a verdict requiring both the sensitivity and the specificity criterion to pass is an intersection–union test, so the specificity criterion cannot raise the probability of wrongly passing the sensitivity claim. No verdict, count or decision rule changes. This milestone appeared while the title still read 1.7.1.

Each milestone builds on the one before; milestones 11, 16 and 17 withdraw earlier claims. The Bernoulli/binomial model of milestone 1 is still the model; since milestone 16, Wilson describes results but no longer decides verdicts.

Forward Scope

The items once targeted for 1.4, deferred from 1.3 because each required coordination with downstream frameworks and fixtures, stand as follows in 1.5.0:

- **1.4-S1**: Baseline indexing carries sampling identity. Open. Under the model of §1.4.2 a baseline is one run over one sampling that all its criteria share, and differing input populations are separate contracts with their own baselines (§1.5.6); what remains is recording that sampling's identifier on the baseline.
- **1.4-S2**: Threshold-notation cleanup separating the Wilson real-valued output, the integer cutoff, the displayed rate, and the compliance requirement. Resolved by milestone 16: no decision rule produces a real-valued Wilson output any more, and the binding artefacts — the regression cutoff c, the compliance count $k_{\min}$, the latency rank — are defined in §3.4, §3.6, §12.4.2 and Appendix A, with the displayed rate c/n_t and the requirement p_{req} kept apart from them.
- **1.4-S3**: Predictive / two-sample treatment of latency regression. Resolved by milestone 16, with `latency/precedence` (§12.4).

Introduction: The Assumption of Certainty in Software Testing

For decades, the dominant paradigm in software testing has rested on an implicit assumption: **systems under test behave with certainty**. Given the same input, a correctly functioning system produces the same output. Tests produce **binary outcomes** — pass or fail — a single failure is definitive evidence of a defect, and uncertainty, when it intrudes, is treated as a ”flaky test” to be mocked away. Continuous integration, test-driven development, and quality gates all sit on that foundation.

Pockets of statistical practice have always existed alongside this mainstream — reliability engineering, performance engineering, queueing analysis, randomised testing, stochastic simulation — but they have sat outside the automated-test workflow rather than inside it. What is new, and what makes the question pressing now, is that **Large Language Models** promote uncertainty from a nuisance to an intrinsic property of the system under test. Identical inputs produce different outputs, by design; the distribution of outputs **is** the behaviour. That promotion makes statistically disciplined repeated-trial testing a first-class concern of mainstream software engineering, not the specialist subfield it used to be.

Traditional Testing	Testing Under Uncertainty
Accidental (bugs)	Intentional (sampling)
To be eliminated	To be characterised
Failure is binary	Failure is probabilistic

Traditional Testing	Testing Under Uncertainty
Single test is definitive	Single test is a sample

Two Dimensions of Stochasticity

Uncertain-system behaviour manifests along two **distinct** quality dimensions:

1. **Functional stochasticity** — whether the system produces a correct result. Given identical input, an LLM may generate valid JSON in 95 out of 100 invocations and malformed output in the remaining 5. Correctness is a random variable.
2. **Temporal stochasticity** — how long the system takes to respond, even among successful invocations. Latency is not a fixed property; it is a distribution.

The two dimensions are distinct (or orthogonal as quality concerns), not necessarily statistically independent. A fast response can be incorrect; a slow response can be correct; but correctness and latency may also covary in practice — challenging prompts may be slower and more likely to fail, overloaded infrastructure may inflate both error and tail-latency rates, and safety filters may change both refusal behaviour and latency. **The methodology does not require functional and temporal stochasticity to be statistically independent; the combined verdict is a logical conjunction rather than a probabilistic independence model.** Both dimensions require repeated observation and distributional reasoning. The mavai methodology treats them with different statistical machinery — a binomial model for functional outcomes (§§1–5), non-parametric empirical percentiles for latency (§12) — and requires both to pass for the overall test to pass.

Within the functional dimension, a contract may declare one or more **criteria** (§1.4) — each its own Bernoulli stream, with its own threshold and error level α. Criteria within a single experiment share the experiment's sampling and so share the N samples posted to the service; evidence about a different input distribution comes from a separate test, bound to its own contract, with its own sampling (§1.4.2). The contracts that evidence one business service over several input populations form a family of contracts; a view across them is reporting (§1.4.7). The per-criterion partition refines the evidence within the functional dimension; it does not introduce new dimensions of stochasticity, since every criterion shares the same methodological regime (the binomial model and its decision rules on per-criterion pass-rates). A parallel diagnosis, with a different methodological response (Generalized Stochastic Dominance), appears in Garces Arias et al. (2025).

Memory consumption, token usage, and cost per call also vary and could be modelled the same way, but the methodology concentrates on correctness and latency because these two dimensions have the most direct impact on end users; resource consumption is usually managed through infrastructure tooling rather than test assertions.

What This Document Contains

The document mixes three kinds of content, and it is useful to name them upfront so the reader knows the epistemic status of each claim:

- **Exact statistical results** — e.g. the Bernoulli/binomial model, the one-sided Fisher exact test that decides empirical regression, the exact one-sided binomial test that decides normative compliance, the Clopper–Pearson bound, and the precedence rank that decides latency regression. These are theorems, cited and verifiable.
- **Operational approximations** — e.g. normal-asymptotic sample-size planning formulas and the Wilson score interval used as the descriptive interval. These are practical approximations chosen for their stability and familiarity; their limits are stated where they appear, and none of them decides a verdict.
- **Engineering guardrails** — e.g. feasibility gates, configuration errors, the distinction between full-strength verification runs and lightweight smoke-test runs, covariate tracking, baseline expiration, transparent-statistics output. These are framework-enforced disciplines that make disciplined inference visible and auditable; they do not replace the underlying statistics.

Where a device below belongs to one of these categories, it is labelled. Readers should not expect design policies to have the calibration guarantees of theorems, or vice versa. Every error-rate statement names the random experiment it calibrates: for empirical regression that experiment draws both the baseline count and the test count afresh under an unchanged success probability (§3.4); a probability that holds the baseline fixed at its observed rate is reported, where it appears at all, only as a labelled diagnostic.

The mavai Project Family

The methodology is implemented across language-native frameworks:

Framework	Language	Role
punit	Java	JUnit -compatible reference implementation
feotest	Rust	Idiomatic Rust implementation (not a Java port)
baseltest	Python	(planned)

Each implements the same statistical core independently. Language-agnostic reference data for conformance testing is generated by the R-based mavai-R project; a per-language implementation is conformant when it reproduces those outputs within stated tolerances. Detailed cross-language conformance architecture is documented in the project READMEs.

Document Purpose and Audience

This document provides a rigorous statistical treatment of the methods employed by the mavai methodology for probabilistic testing of systems characterised by uncertainty. It is intended for:

- **Professional statisticians** validating the mathematical foundations
- **Quality engineers** with statistical training designing test strategies
- **Auditors** who need to verify that testing methodology is sound
- **Technical leads** establishing organizational testing standards
- **Framework implementors** building mavai-compatible statistics engines in new languages

---

The Model in Brief: Contracts, Tests, Experiments

The rest of this document uses a small vocabulary — service contract, criterion, test, threshold, experiment, verdict — that it will lean on heavily. The terms are not exotic, but each carries a precise meaning in this methodology, and the chapters that follow are easier to navigate when they are introduced together.

The foundation is the **service contract**. A service contract is the definition of what a correct response from the service under test looks like. It is not a single yes/no proposition; it is a conjunction of separately checkable expectations, and each expectation is called a **criterion**. A criterion asks one yes/no question of a single response: did this output meet this expectation, or did it not? A contract typically declares several criteria — schema validity, semantic correctness, absence of disallowed content — and the framework evaluates every one of them against every response it observes.

A **test**, in this document, is what a developer or a CI system invokes to check that the service is behaving acceptably. Because the service is stochastic, a single response cannot settle the question; a test therefore invokes the service many times, applies the contract's criteria to each response, and aggregates the per-criterion outcomes into a **pass rate**. The test then compares that pass rate against a **threshold** and reports a per-criterion **verdict**: PASS, FAIL, or INCONCLUSIVE.

A criterion's threshold comes from one of two places. A **normative** threshold is a fixed value the contract is told to meet — typically by an SLA, an SLO, a policy, or a regulation. An **empirical** threshold is derived from the service's own measured behaviour: the test does not ask whether the service meets some externally mandated rate, but whether it has degraded from a baseline it previously achieved. Empirical thresholds require a baseline, and a baseline is produced by an **experiment**: a separate invocation of the service whose purpose is to record the rate at which the service currently satisfies each criterion. Tests configured with normative thresholds do not need an experiment; tests configured with empirical thresholds consult the most recent experiment's baseline.

The remaining vocabulary is structural. Whether the invocations belong to a test or to an experiment, the list of inputs the framework executes against the service is called a **sampling** — a list of $N \geq 1$ inputs whose responses are scored against every criterion in the contract.

The bulk of this document is, in the end, an account of how those verdicts are computed under each threshold type, what they entitle a reader to claim, and what disciplines keep them honest. With that vocabulary in hand, the document can now look

more closely at the kinds of expectation a contract can carry — which is the subject of the next section.

Clause Forms: Rate-Bounded and Categorical

A service contract is a conjunction of clauses, but the clauses are not all of one kind. Two structurally different forms coexist in a typical contract, and the rest of this document depends on keeping them apart. The distinction is orthogonal to the threshold-origin axis the companion already carries (compliance vs. regression, §3): a clause can be normative or empirical in origin — mandated by SLA/SLO/policy/regulation, or derived from a measured baseline — independently of the form the present section partitions on.

A **rate-bounded clause** states a rate-bounded proposition: "the service shall satisfy criterion c at rate at least p_c^* with confidence γ over sampling V." Its threshold p_c^* has one of two **origins** — **normative** (a.k.a. stipulated), an SLA-mandated rate evaluated by the compliance paradigm of §3, or **empirical**, a measured baseline evaluated by the regression paradigm of §3. The word empirical names that origin and is **not** the name of this form; compliance and regression are simply the decision-side names for the two origins. Either way the form is rate-bounded: a test invokes the service many times, and the companion's central machinery turns the resulting evidence into a per-criterion verdict.

A **categorical clause** states an obligation or prohibition that admits no rate threshold at all: "the service shall not violate criterion c." Full stop. The clause is satisfied or violated in kind, not in rate. Categorical clauses are, in practice, almost always normative in origin — a categorical prohibition is the kind of thing a contract mandates rather than something a team derives from baseline measurement — but their distinguishing feature is form rather than origin: no p_c^*, no γ, no Wilson construction, no rate-bounded discharge available. The canonical examples are categorical safety prohibitions: no self-harm advice, no PII leakage, no emission of illegal content.

Rate-bounded and categorical clauses are not two points on a strictness spectrum. Letting $p_c^* \to 1$ does not promote a rate-bounded clause into a categorical one: it merely tightens the rate-bound proposition. No sample budget, threshold, or confidence level redenominates a categorical claim as a frequentist one — observing zero failures in n trials yields, at best, a rule-of-three upper bound of $\approx 3/n$ on the failure rate, which approaches zero only in the limit. A categorical clause therefore cannot be discharged by the rate-bounded apparatus, however that apparatus is parameterised.

A categorical clause can be stated on any contract, and wherever it is stated it is evaluated observationally (§1.4.5): one observed violation falsifies it, and a run without one shows only that none was seen. Where the service cannot realistically be made to produce the prohibited content often enough to establish its reliability, a separate component — a guardrail, a deterministic filter, a hard schema constraint, a refusal classifier — may be interposed between the stochastic system and the consumer of its output, and tested in the service's place; that architectural pattern is the subject of Distributional Contracts (§4.6, §8.9), not of this document. Statistically,

the component is itself probabilistic; its performance enters as one or more **ordinary rate-bounded clauses**, typically covering its false-negative behaviour over an adversarial sampling of the content it exists to filter and its false-positive behaviour over a benign one. The clause form is fixed — rate-bounded — but the threshold origin is free: it may be **normative**, where a safety policy, regulator, SLO, or release gate mandates a minimum containment level, or **empirical**, where the clause is used for regression against a measured guardrail baseline. These clauses evidence the component; the categorical clause remains an obligation, evaluated observationally wherever it is stated.

The methodology summarised across the rest of this document is therefore:

> Tolerable failures are bounded statistically; intolerable failures are prohibited categorically; and what cannot be tested in the service is tested in its guardrail.

The rate-bounded/categorical distinction is foundational. The compliance/regression paradigms of the next section, the criterion decomposition of §1.4, and the report and verdict layers of §7 and §10 all operate within the rate-bounded class — across both normative-origin and empirically-derived thresholds. The statistics of the guardrail pattern are developed in §1.4.5b. Its contractual side — how a commitment is declared and what its record holds — is not this document's subject: it belongs to Distributional Contracts, whose chapter on composition sets it out (§8.9).

Two Testing Paradigms: Compliance and Regression

Within the rate-bounded class of contract clauses introduced in the previous section, the methodology supports two distinct testing paradigms. They share the same hypothesis-test skeleton but differ in where the threshold comes from and how results are interpreted.

Paradigm	Threshold source	Statistical question	Example
Compliance	Contract / SLA / SLO / policy — given, not estimated	”Does the system meet its mandated requirement?”	Payment API with contractual $p_{\text{SLA}} = 0.995$ uptime
Regression	Empirical estimate from a MEASURE experiment	”Has the system degraded from its measured baseline?”	LLM customer-service system with $\hat{p}_{\text{baseline}} = 0.951$ from $n = 1000$ samples

Both paradigms share a one-sided binomial decision skeleton, but they implement **two related but distinct procedures with different decision semantics and different error interpretations**. The methodology treats them as separate inferential acts: the controlled error rate, the direction of evidence, and the meaning of a PASS

or FAIL differ between the two, and reports name which procedure produced a given verdict, by its versioned identifier (§7, §10).

Compliance / assurance procedure (`compliance/exact-binomial`) — affirmative procedure used to show that a system meets an externally required threshold p_{req}:

$$H_0 : p \leq p_{\text{req}} \quad \text{(not shown acceptable)} \qquad H_1 : p > p_{\text{req}} \quad \text{(shown acceptable)}$$

PASS iff the observed count K reaches the smallest count the exact one-sided binomial test accepts, $k_{\min} = \min\{k : P_{p_{\text{req}}}(K \geq k) \leq \alpha\}$ — equivalently, iff the one-sided Clopper–Pearson lower bound on p is at least p_{req} (§3.6). The probability of falsely declaring compliance when the true success probability is at or below the requirement is at most α, exactly, at every sample size.

Regression / monitoring procedure (`regression/fisher`) — procedure used to detect degradation from a measured baseline of K_b successes in n_b trials. The hypotheses compare the test's success probability p_t with the baseline's p_b:

$$H_0 : p_t \geq p_b \quad \text{(no degradation)} \qquad H_1 : p_t < p_b \quad \text{(degradation)}$$

The integer cutoff c is derived from (K_b, n_b, n_t, α) by the one-sided Fisher exact test (§3.4). PASS if the observed test count $K_t \geq c$; FAIL if $K_t < c$. With the baseline count and the test count both random under an unchanged success probability, the probability of a false degradation signal is at most α. **A regression PASS means "no degradation signal at this cutoff," not "equivalence to the baseline has been proved"**, and a compliance PASS means "evidence supports compliance at the configured level," not "no degradation."

The two rules share the binomial model, the integer-valued decision artefact and the report structure, but their PASS/FAIL labels are **not interchangeable**. The transparent-statistics output (§7, §10) records which procedure was applied, and reports use procedure-specific verdict wording so that auditors and operators do not read a regression PASS as an affirmative compliance certification (or vice versa).

The regression-side prose of §§3–5 also uses a displayed rate: c/n_t, the rate corresponding to the integer cutoff. The displayed rate is informational; the binding decision artefact is c (§3.4, §5.1).

The three remaining operational differences are the **source** of the threshold (given vs. derived), the **interpretation** of failure (requirement not demonstrated vs. degradation signal), and the **prerequisite step** (none vs. MEASURE).

Two configurations are refused before any sample runs (§5.7.1): a compliance design too small for PASS to be possible, under VERIFICATION intent, and a test planned larger than the baseline it consumes.

A requirement and a baseline on the same postconditions. A service may be judged against a contractual requirement and against its baseline at once, and the two answer different questions: the requirement asks whether the service is good enough, the baseline whether it has degraded since it was measured. A service can

meet its requirement with room to spare while its baseline shows it slipping, and the reverse. The two questions are two criteria over the same postconditions: a compliance criterion, decided by `compliance/exact-binomial` against p_{req}, and a regression criterion, decided by `regression/fisher` against the baseline. Each has one rule, one α, one threshold origin and one verdict, and both evaluate the same observations of the same run. The test's verdict is the ordinary structural composite of §1.4.6 — FAIL if either criterion fails, naming it — and the two criteria enter the two error envelopes by their directions. A test carrying both is refused whole if either part is invalid (§5.7.1).

---

1. Statistical Model

1.1 Bernoulli Trial Framework

Each invocation of the service contract is treated as a Bernoulli trial under a **working approximation** of independence and stationarity:

$$X_i \sim \text{Bernoulli}(p)$$

where:

- $X_i \in \{0, 1\}$ is the outcome of the i-th trial (1 = success, 0 = failure)
- $p \in [0, 1]$ is the true (unknown) success probability
- Trials are assumed independent and identically distributed (i.i.d.)

The word working is load-bearing: independence and stationarity are modelling assumptions that hold approximately in practice, not discovered truths about the system under test. §1.3.1 sets out why the assumption is nevertheless defensible as a starting point for a typical LLM experiment; §1.3.2 separates the formal assumptions from the operational threats to those assumptions; §8 describes the framework-level guardrails that make assumption drift visible without pretending to repair it.

1.2 Binomial Aggregation

For n independent trials, the total number of successes follows a binomial distribution:

$$K = \sum_{i=1}^{n} X_i \sim \text{Binomial}(n, p)$$

The sample proportion $\hat{p} = K/n$ is an unbiased estimator of p:

$$E[\hat{p}] = p, \quad \text{Var}(\hat{p}) = \frac{p(1-p)}{n}$$

1.3 Assumptions and Limitations

The Bernoulli model is a working approximation, not a discovered truth. §1.3.1 sets out why the assumption is reasonable for a typical LLM experiment; §1.3.2 enumerates the formal assumptions and the operational threats to each; §1.3.3 describes the developer's role in upholding the independence premise.

1.3.1 Why the Working Approximation is Defensible A natural objection to the Bernoulli model for LLM testing is that an LLM is not a coin: its outputs depend on weights that the test author neither owns nor controls, served by infrastructure that the test author cannot inspect. Why should the success probability p be treated as constant within an experimental run, given that something about the served system can change at any time?

The answer rests on a layered view of what determines a trial's outcome. In decreasing order of magnitude, a verdict is shaped by:

1. **Model weights** — the parameters that define the model's capabilities and dispositions.
2. **Serving stack** — quantisation, kernel choices, batching, speculative decoding, mixture-of-experts routing.
3. **Sampling configuration** — temperature, top-p, max tokens, set by the caller.
4. **Numerical realisation** — floating-point ordering on the inference hardware, sensitive to batch composition.
5. **Input distribution** — the prompts and contract context fed to the model.

Layer 1 is the dominant term and is the layer for which the assumption of stability is most defensible. Major LLM providers commit, as a matter of policy, that their model identifiers resolve to immutable weights for the lifetime of the model's deprecation window. Anthropic's documentation states this explicitly: ”Every Claude model ID is a pinned snapshot. Models with a date in the ID (for example, *20250929*) are fixed to that specific release. Starting with the Claude 4.6 generation, model IDs use a dateless format that is also a pinned snapshot, not an evergreen pointer” (Anthropic, 2026). For older generations, providers also expose **floating aliases** — convenience pointers that resolve to the current dated snapshot at call time and may be re-bound when a newer snapshot ships. Calling through a pinned snapshot fixes layer 1 across the experimental run by provider commitment; calling through a floating alias does not.

Empirical study has documented material drift in model behaviour over months when calls are routed through floating aliases (Chen, Zaharia & Zou, 2023). The methodology absorbs this finding: the prescribed posture is to test against pinned snapshots and to set explicit baseline-validity windows (§8.4.2) that require operators to reconfirm p before stale comparisons are drawn.

Pinned model IDs are necessary but not sufficient. Pinning the model identifier reduces one major source of non-stationarity — weight changes — but does not by itself guarantee stationarity of the full served system. Layers 2–5 above remain potential sources of drift even when layer 1 is fixed: serving-stack revisions (kernel updates, quantisation changes, batching policy, speculative-decoding configuration), routing

changes (mixture-of-experts gating, regional load-balancing, edge-cache topology), safety-layer revisions (prompt classifiers, output filters, refusal heuristics), regional deployment variation (different provider regions running different stack revisions concurrently), and numerical-kernel changes can each shift observed behaviour while the model ID remains the same string. Provider documentation describes endpoint and routing configurations that vary across platforms (Anthropic enterprise documentation, accessed 2026-05-14), reinforcing the need to capture endpoint and deployment metadata. The factor record (§1.5.2) therefore captures all observable service, endpoint, prompt, sampling, guardrail, and deployment metadata available to the test operator — not the model ID alone — and the guardrail severity policy of §8.4.5 treats endpoint and serving-stack changes as major caveats under VERIFICATION.

Layers 2 and 4 are the source of the well-documented residual non-determinism observed even at temperature zero with fixed sampling configuration: the same prompt to the same snapshot can yield different outputs in successive calls. This affects the variance of the trial outcome — exactly the phenomenon the binomial model is built to absorb — without changing p in the population sense the model requires. Provided the input distribution (layer 5) is also stable across the run, and the inputs are independent draws from a stable generator, the verdict sequence is well-modelled as a sequence of Bernoulli trials with constant p. A fixed list of prompts, invoked once each, is different: its outcomes are independent but not identically distributed, $X_j \sim \text{Bernoulli}(\theta_j)$ with a success probability per prompt, and their total is Poisson-binomial given the list. Binomial inference then holds only unconditionally over how the list was drawn — if the list is an i.i.d. sample from a named prompt population — and a list reused across runs is a different design again (§3.4, §8.4.6).

The assumption is therefore defensible under stated conditions: a pinned model snapshot, a fixed system prompt, a fixed sampling configuration, no conversation state carried between calls, and a stable input-sampling process, all within a single experimental run of bounded wall-clock duration. Outside these conditions — most importantly, calls through a floating alias, or experiments whose duration spans a known provider rollout window — the assumption can fail, and the diagnostics of §8.3 and the guardrails of §8.4 exist for exactly that case.

1.3.2 Formal Assumptions and Operational Threats The Bernoulli model rests on three **formal assumptions**, each paired with the **operational threats** that could silently violate it in practice:

1. **Independence**: Each trial is independent. In practice, this may be violated if, for example:
    - The LLM provider implements request-level caching
    - Rate limiting causes correlated delays
    - Model state persists across requests (generally not the case for stateless APIs)
2. **Stationarity**: The success probability p is constant across trials. This may be violated if, for example:
    - A baseline is created at a time when the system was under load, but a test

performed later, while the system IS NOT under load. Such a test will likely miss a drop in performance of the system.
- Contextual factors differ between baseline creation and test execution (time of day, day of week, deployment region, etc.)

Non-stationarity is treated in depth in §8.3; the methodology's guardrails for it — covariate tracking (§8.4.1) and baseline expiration (§8.4.2) — do not guarantee stationarity (that is impossible in practice) but make its violation **visible and auditable** rather than silently undermining inference.

3. **Binary outcomes**: Each trial has exactly two outcomes. Complex quality metrics may require more sophisticated models.

1.3.3 Developer Responsibility for Trial Independence While the framework provides the statistical machinery, **developers share responsibility** for ensuring that the independence assumption holds in practice. Many LLM service providers offer features that can introduce correlation between trials:

Feature	Risk to Independence	Mitigation
Cached system prompts	Cached prompts may produce more consistent outputs	Disable caching during experiments, or document its presence
Conversation context	Prior exchanges influence subsequent responses	Use fresh sessions for each trial
Request batching	Batched requests may share internal state	Submit requests individually
Seed parameters	Fixed seeds produce identical outputs	Use random or rotating seeds

Example: An LLM provider's ”cache system prompt” feature improves latency by reusing parsed prompts. While beneficial in production, this can reduce output variance during testing. The developer should either:

- Disable the cache via configuration during experiments
- Document that the measured success rate reflects cached behaviour
- Understand that the true variance may be higher than observed

No framework can detect or correct for these effects—they require domain knowledge and deliberate configuration choices. When trial independence is uncertain, developers should consider:

- Running sensitivity analyses with different configurations
- Documenting assumptions in test annotations or comments
- Increasing sample sizes to account for potential correlation (see Section 8.2)

Recommendation: For most LLM-based systems accessed via stateless APIs, the independence assumption is reasonable when caching and context features are disabled. Monitor for temporal drift in long-running experiments.

1.4 Criterion Decomposition of the Functional Dimension

1.4.1 Failure modes of differing kind Criterion decomposition operates within the functional dimension introduced earlier in the document. The partition this chapter develops refines the evidence inside that one dimension into multiple statistical streams; it does not multiply the number of dimensions of stochasticity.

The chapter uses a deliberately stark **triple** as its running example, drawn from a clinical-advice service:

- a structural failure — a response that does not parse as JSON;
- a judge-mediated failure — a response that parses, but whose register is wrong for the apparent literacy of the recipient (clinical jargon to a layperson, say); and
- a catastrophic failure — a response that advises a vulnerable user toward self-harm.

The three modes are not interchangeable, and each carries a distinct methodological load. Parseability and register are both rate-bounded criteria — they share clause type despite the gulf between their measurement apparatus (a JSON parser, a rubric-driven judge), and together they establish that the rate-bounded machinery of this chapter applies as readily to judge-mediated criteria as to mechanical ones. Self-harm is categorical rather than rate-bounded: it sits in a different clause type despite, like register, requiring a judge to detect, and the clause-type partition therefore does not run along the mechanical/judge-mediated axis. References to self-harm in what follows are technical illustrations of a categorical clause; they are not commentary on clinical practice or on the regulation of clinical AI systems.

A service contract's postconditions defend against failure modes that vary along at least three independent axes. Each axis is on its own sufficient to require separate statistical treatment of failure modes that differ along it, and each — at its extreme — also reveals the boundary between the rate-bounded and categorical clause classes introduced in Clause Forms: Rate-Bounded and Categorical.

Consequence. Parseability, register, and harm all violate the contract, but the cost of each violation to the consumer of the verdict spans a wide gradient, and the methodology distinguishes points on that gradient at two different levels. Within the rate-bounded class, where the cost is bounded and a non-zero failure rate is tolerable, consequence is operationalised as the per-criterion threshold p_c^*: register (medium-consequence) demands a tighter p_c^* than parseability (low-consequence), and both are evaluated by the same binomial decision machinery despite the gulf between their measurement apparatus (a JSON parser, a rubric-driven judge). Across the rate-bounded/categorical boundary, where the project's stance is zero tolerance — harm advice, PII leakage, illegal content — consequence is no longer operationalised as a tight p_c^* at all. The failure mode is admitted as a categorical clause and evaluated in the observational mode of §1.4.5, which is not a limit-case of a rate-bounded clause at $p_c^* \to 1$. Where the service cannot be tested for it, a guardrail is the pattern (Distributional Contracts §8.9), evidenced first by its own rate-bounded contract (§1.4.5b).

Frequency. Baselines of rate-bounded criteria span a wide range — parse-failure rates in the low single percents, register-mismatch rates higher still, other rate-

bounded criteria potentially lower. The sample budget required to support strong evidential power scales inversely with the baseline, and rate-bounded peers with disparate baselines therefore make competing demands on the experiment's shared sampling. A thousand-sample sampling gives an interval around a 5%-baseline observation tight enough to support a binding evidential claim against a nearby threshold; as the baseline falls, the same pool gives an interval whose width relative to the observed rate grows — at a 0.1% baseline the Wilson interval spans several times the point estimate itself — and a criterion at that baseline either needs a larger pool to be binding under VERIFICATION intent, or must be declared SMOKE up front. Per-criterion feasibility gates surface this asymmetry honestly, where an aggregated stream absorbs it. The categorical clause sits off this axis: a 'zero-failures' failure mode is not admitted to the contract as a rate-bounded criterion at all, and so has no baseline to enter the comparison.

Input requirement. A MEASURE experiment runs over a single sampling: every sample produces a response, and every criterion evaluates its postcondition(s) on the same response. The criteria share the input distribution because they share the inputs themselves. A sampling sized for parseability serves register and other rate-bounded judge-mediated criteria without trouble, because their baselines are commensurate. What it cannot do is establish a 'zero-failures' failure mode: the rule-of-three bound on the failure rate with zero observed failures is $\approx 3/n$, which approaches zero only in the limit, and no n reachable inside one experiment closes the gap. A categorical clause stated on the contract is therefore evaluated observationally (§1.4.5): one observed violation falsifies it, and a clean run reports only that none was seen. Where the service cannot realistically be made to produce the prohibited content often enough to say more, a guardrail can be tested in its place, by a separate test bound to its own contract over an adversarial sampling (§1.4.5b; see Clause Forms: Rate-Bounded and Categorical).

The methodology therefore partitions on two levels. Within the rate-bounded class, failure modes that differ along consequence, frequency, or input share are treated as **separately contractual** — each is its own hypothesis test, with its own threshold, its own error level α, its own feasibility gate, sharing the experiment's sampling. Across the rate-bounded/categorical boundary, a failure mode is stated as a categorical clause and evaluated observationally, in the MEASURE experiment or in any other test whose contract states it, with the epistemic status set out in §1.4.5; where the service cannot be tested for it, a guardrail carries the rate-bounded evidence in a separate test (§1.4.5b).

A representative example, threaded through the chapter, is a clinical-advice service whose contract carries:

- P_1: response parses as JSON — rate-bounded, low-consequence (parser-detected).
- P_2: required fields are present — rate-bounded, low-consequence (parser-detected).
- P_3: response is layperson-readable — rate-bounded, medium-consequence (judge-detected).
- P_4: advice does not suggest self-harm — **categorical**, catastrophic-consequence. Evaluated observationally. Because the model cannot realistically be coaxed into

producing self-harm advice often enough to establish its reliability, a dedicated guardrail component (e.g., a harm classifier interposed between the model and the user) is tested in its place; the service's behaviour over an adversarial sampling is reported as an observational zero-failures criterion in a separate test, and P_4 may additionally be evaluated on the production sampling (§1.4.5).

The four postconditions are not interchangeable along the partition axes. A P_4 violation at any rate is clinically significant in a way a P_1 violation at 10^{-3} is not, and a P_3 violation at 5% is significant in a way a P_1 violation at 5% is not. The methodology evaluates the rate-bounded postconditions (P_1, P_2, P_3) each in its own statistical stream, against its own threshold, at its own error level: within the primary end-to-end MEASURE experiment where they share its input population, and in a separate test with its own sampling where a postcondition is to be evidenced over a designed input population, as P_3 is in §1.4.8. P_4 is evaluated observationally wherever it is stated; the guardrail that polices it carries its own rate-bounded criteria (normative or empirical in origin), evaluated in separate tests over adversarial and benign samplings (§1.4.5b, §1.4.8). The three primitives of §1.4.2 give the empirical partition its formal structure; the hiding result of §1.4.4 establishes that an aggregated stream over rate-bounded postconditions that differ along any of the three axes potentially obscures movement in a low-frequency, high-consequence, or designed-input criterion. Where rate-bounded postconditions defend against failure modes that are interchangeable along all three axes — equivalent consequences, comparable frequencies, the same input distribution — a single aggregated stream remains an adequate representation; the $m = 1$ instance of the per-criterion model recovers this case unchanged.

1.4.2 Three primitives The decomposition rests on three primitives, each playing a distinct role in the statistical model.

Sampling. The list of sample inputs posted to the service under test in a single experiment or test. A sampling has length $N \geq 1$; each sample entry in the list is presented once to the service, producing N responses. The sampling is shared across every criterion (defined below) that the contract exercises in the run: a contract with multiple criteria produces a per-trial vector of per-criterion observations over a single shared sampling, and there is no notion of a ”per-criterion sampling” within an experiment. One run has one sampling, and every criterion of its contract shares it. Evidence about a different input distribution comes from a separate test, bound to its own contract, with its own sampling. The inferential reach of a sampling is elaborated in §1.4.7.

Criterion. The unit of statistical evaluation. A criterion is the partition unit of the functional dimension; each criterion is exercised against the sampling and yields its own sequence of per-trial pass/fail outcomes, modelled as a Bernoulli stream (§1.4.3), from which its own verdict is computed. A criterion declares the mode under which the test is conducted (inferential or observational, §1.4.5), the threshold and threshold origin where applicable, the error (significance) level α, and the experiment or test in which the criterion is exercised (§1.4.7). It also hosts one or more postconditions (defined below), which together determine its per-trial outcome: a single-

postcondition criterion produces a per-trial outcome equal to that postcondition's verdict; a multi-postcondition criterion produces a per-trial outcome equal to the conjunction of its hosted postconditions' verdicts (§1.4.3 makes this formal). Two postconditions whose failures carry materially different consequences are therefore addressed by two distinct criteria, never by sharing a stream.

Postcondition. A named predicate over the produced output of a single trial. A postcondition has one job: decide pass or fail for a single observable property of the output. It carries no threshold and no statistical configuration of its own; the threshold, confidence level, and mode under which its per-trial verdicts are aggregated come from the criterion that hosts it. In the clinical-advice example, P_1 through P_4 above are postconditions.

The relationship between the primitives is:

Primitive	Role
Sampling	List of $N \geq 1$ samples posted to the service in one experiment or test; shared by all criteria of the run
Criterion	Statistical partition unit; exercised against the sampling; hosts 1+ postconditions
Postcondition	Per-trial predicate; hosted by a criterion

Each of the three primitives has exactly one job: the sampling supplies the N samples the run posts to the service and over which every criterion is exercised; criteria partition the functional dimension into statistical streams and parameterise each stream's inferential test; postconditions decide per-trial outcomes.

1.4.3 The per-criterion Bernoulli model Each criterion c defines a Bernoulli stream of per-trial pass/fail indicators $\{X_{i,c}\}$, aggregated to a Binomial success count $K_c = \sum_i X_{i,c}$ for inference; the per-stream parameter is p_c and the per-stream estimate is $\hat{p}_c = K_c/n_c$. A contract with a single criterion ($m = 1$) is the special case of this formulation in which the per-criterion stream coincides with the single Bernoulli/Binomial model of §§1.1–1.2; the multi-criterion machinery below covers $m \geq 1$ uniformly.

Let a contract declare m criteria $\{C_1, \ldots, C_m\}$ (the symbol m for the number of criteria is used throughout this chapter and the rest of the companion; K is reserved for the **success count** of a Bernoulli stream as in §§1.1–1.2 — $K = \sum_i X_i$ — so the two must not collide). For each criterion c, let $\mathcal{P}_c$ denote the set of postconditions the criterion references. Let n_c denote the number of trials in the experiment's sampling; every sample is presented to every criterion, so n_c equals the sampling size N (§1.4.5a). On each trial define the per-criterion observation:

$$X_{i,c} = \begin{cases} 1 & \text{if } c \text{ produced a testable value and every } P \in \mathcal{P}_c \text{ holds on trial } i \\ 0 & \text{otherwise.} \end{cases}$$

A trial scores 0 both when the postcondition was evaluated and did not hold and when no testable value could be produced (a transform/no-value FAIL, §1.4.5a); the two are distinguished by a diagnostic reason, not in the arithmetic. The effective denominator is $n_c = N$ — every trial of the sampling — and the success count is $K_c = \sum_i X_{i,c}$, so $\hat{p}_c = K_c / n_c$.

Each per-criterion trial is modelled exactly as the single-criterion trial of §§1.1–1.2:

$$X_{i,c} \sim \text{Bernoulli}(p_c), \qquad K_c = \sum_{i=1}^{n_c} X_{i,c} \sim \text{Binomial}(n_c, p_c), \qquad \hat{p}_c = K_c / n_c.$$

The decision rules of §3 (`regression/fisher` and `compliance/exact-binomial`), the feasibility gate, the configuration errors and the verdict-evaluation rule apply to this effective stream.

The independence and stationarity working approximations of §1.3 apply per criterion. Within a criterion, the per-trial observations across the experiment's N samples are treated as i.i.d. under the input distribution the sampling represents. Across criteria, the methodology does not assume independence: the per-criterion trials $X_{i,1}, \dots, X_{i,m}$ on a single invocation are typically correlated, since an invocation that produces a malformed response may also fail criteria that evaluate the response's content. Cross-criterion dependence does not affect per-criterion verdict correctness; its consequence for the composite verdict is the disclosed Type-I envelope of §1.4.6.

Each per-criterion trial is a complete statistical object in its own right. From here forward in the companion, the single-criterion X_i, $\hat{p}$, α, p^* of §§1.1–1.2 may be read as the per-criterion $X_{i,c}$, $\hat{p}_c$, α_c, p^*_c for any criterion c declared on a contract: the decision rules, the cutoff derivation, the sample-size derivations, the feasibility gate, and the verdict-evaluation rule apply unchanged.

1.4.4 Why aggregation masks per-criterion failures Aggregating per-criterion indicators into a single conjunction rate is the obvious temptation — one stream, one p, one decision — and this section sets out why it is the wrong move. A one-line union-bound argument shows that the aggregate rate is bounded above by the sum of per-criterion failure rates and dominated by the largest of them; small but consequential per-criterion failure rates are absorbed into the noise of the dominant ones and become statistically invisible. Per-criterion partitioning is what exposes them; the conjunction view buries them.

Let $\{C_1, \dots, C_m\}$ be the contract's criteria, with per-trial indicators $X_{i,c}$ as in §1.4.3. Define the conjunction indicator

$$X_i = \prod_{c=1}^{m} X_{i,c}, \qquad p = \mathbb{P}(X_i = 1)$$

— the rate at which every criterion's indicator simultaneously equals one on a trial. By the union bound applied to the complementary events,

$$1 - p \;=\; \mathbb{P}(\exists\, c : X_{i,c} = 0) \;\leq\; \sum_{c=1}^{m} (1 - p_c)$$

with equality if and only if the per-criterion failure events are disjoint. The aggregate failure rate $1 - p$ is therefore bounded above by the sum of per-criterion failure rates, and in typical cases is dominated by the largest of them.

Corollary. If criterion c^* has a per-criterion failure rate that is small relative to the largest per-criterion failure rate — say $1 - p_{c^*} \ll \max_{c \neq c^*}(1 - p_c)$ — then $1 - p$ is dominated by the larger terms and is essentially insensitive to $1 - p_{c^*}$. A change in $1 - p_{c^*}$ of any magnitude substantially smaller than $\max_{c \neq c^*}(1 - p_c)$ produces a change in $1 - p$ within the sampling noise of an aggregate estimator. A conjunction indicator therefore cannot detect movement in $1 - p_{c^*}$ unless that movement is comparable to the noise of the dominant criterion. The masking mechanism is purely a property of frequency: rare per-criterion failure rates are absorbed by more frequent ones, regardless of what those failures mean. The reason this matters for asymmetric contracts is empirical, not constructive: catastrophic-consequence outcomes (a model that recommends self-harm, a payment service that double-charges, a clinical-advice system that produces an unsafe instruction) are, in practice, rare, often deliberately so — modern LLM safeguards, hard-coded blocklists, and validation layers exist precisely to drive their rate toward zero. Rarity is what makes them mathematically invisible to aggregation; gravity is what makes that invisibility intolerable.

Inverse statement. Given an observed conjunction rate $\hat{p}$, the per-criterion rates $\{\hat{p}_c\}$ are not identified: any allocation of $1 - \hat{p}$ among the per-criterion failure events that respects the union bound is consistent with the observation. Recovering the per-criterion rates requires per-criterion attribution at trial record time — the wide trial record that §1.4.3's indicators demand. A trial archive that has not preserved per-postcondition outcomes per trial cannot be decomposed retrospectively.

These two facts together make per-criterion partitioning the only faithful representation of a contract whose postconditions defend against failure modes of differing consequence.

1.4.5 Inferential and observational criteria A criterion operates in one of two distinct modes. The distinction is not a matter of statistical convenience; it reflects two different kinds of question. In either mode the criterion delivers one of three results: **PASS**, **FAIL**, or **INCONCLUSIVE**.

Inferential criterion. Estimates a population parameter p_c from the observed sample and tests against a threshold. The threshold p_c^* is either contractual (origin SLA, SLO, POLICY) or empirically derived (origin EMPIRICAL). The verdict follows a procedure-direction-specific decision rule (§3.2, §5.1): a **compliance** criterion issues PASS iff the observed success count K_c reaches $k_{\min}$, the smallest count the exact

one-sided binomial test accepts at α_c (§3.6); a **regression** criterion issues PASS iff K_c meets or exceeds the integer cutoff c_c derived from the baseline's counts by the one-sided Fisher exact test at α_c (§3.4). Where no verdict can be rendered — $n_c = 0$, or no count of the observed sample could pass the compliance test (§5.7.1) — the verdict is INCONCLUSIVE; configurations refused before the run carry a configuration error instead (§5.7.1). The inferential mode is the appropriate choice for criteria whose contractual question is "what is the true rate of behaviour c, with what confidence, and does it clear the demanded threshold?".

Observational criterion. Reports whether any failure of the criterion was observed in the run. No population estimation; no confidence interval; no threshold parameter. The test is on the observation itself:

$$\text{verdict}(c) \;=\; \begin{cases} \text{PASS} & \text{if } K_c = n_c \text{ and } n_c > 0 \\ \text{FAIL} & \text{if } K_c < n_c \\ \text{INCONCLUSIVE} & \text{if } n_c = 0. \end{cases}$$

A PASS verdict says exactly: zero failures of c were observed in the n_c trials of the experiment's sampling. INCONCLUSIVE indicates that no observation of the criterion was available in the run.

In this formula n_c is the criterion's number of trials, equal to the sampling size N (§1.4.5a). An observational (`zeroFailures`) criterion estimates no proportion and so carries no rate denominator: it is PASS when no failure is observed across its trials, FAIL on any observed failure (including a transform/no-value failure), and INCONCLUSIVE only when there were no trials ($n_c = 0$).

Exact. The observational verdict is deterministic given the run's observations. It makes no claim about p_c. A passing observational verdict at $n_c = 1000$ means exactly: no failure of criterion c was observed in 1000 trials of the experiment's sampling. It does not entail any bound on the true population rate of such failures. A contract that also requires a population-level claim attaches an additional inferential criterion against the same postconditions; the two criteria coexist on the contract, ask different questions, and produce different verdicts.

Engineering guardrail. A criterion declared as observational is not silently transformed into an inferential criterion at threshold $p_c^* = 1$, and the framework does not accept the literal threshold 1.0 as an inferential parameter. Under $p_{\text{req}} = 1$ even n_c successes in n_c trials have probability 1, above every $\alpha_c < 1$; an inferential test against $p_c^* = 1$ cannot pass at any finite sample size. The observational mode is the honest expression of zero-failures contracts: the verdict reports the observation, and the population claim is not asserted from the observation alone; where one is required it is supported architecturally by the guardrail-validation pattern (§1.4.5b), or by separate production monitoring, which this methodology does not currently develop.

Epistemic status of observational mode. What an observational verdict means depends on the clause it serves (see Clause Forms: Rate-Bounded and Categorical). The methodology recognises three uses:

1. Architectural-commitment evidence, complementary. The verdict evaluates an observational zero-failures criterion on an architectural component — typically a guardrail's false-negative behaviour over an adversarial sampling — and the rule-of-three annotation above quantifies it. It complements, and does not replace, the component's own rate-bounded contract, which leads: the guardrail's miss rate over a harmful-candidate sampling, decided exactly (§1.4.5b, Clause Forms). The report reads such a verdict alongside the architectural commitment it evidences (§7, §10).
2. Diagnostic, complementary. The verdict scores the categorical postcondition against the experiment's sampling as a SMOKE-intent criterion. The response is already produced and the judge is already running, so the cost is incidental. The verdict does not establish the categorical clause — the rule-of-three bound on a representative sampling sits orders of magnitude above any zero-failures rate — but it answers a question the adversarial-sampling evaluation cannot: do failures appear in representative production traffic at all? A non-zero result is loud, because the rest of the architecture should be suppressing it. Recommended alongside any guardrail.
3. Without a guardrail. The verdict evaluates a categorical clause stated on a contract with no guardrail behind it. It is the same evaluation as use 2 and carries the same meaning: falsifying evidence when a violation is observed, never establishment when none is. Nothing is missing from the contract; whether the service also needs a guardrail is a separate, architectural question.

The pre-existing mechanics — the literal-1.0 prohibition on inferential thresholds, the NO FAILURE OBSERVED relabel, the rule-of-three annotation — apply across all three uses.

The two modes coexist within a single contract. The clinical-advice service of §1.4.8 uses both, across the three contracts that evidence it over three input populations: an observational criterion against P_4 over adversarial probes, an inferential criterion against P_3 at $p^*_{P_3} = 0.98$, $\alpha = 0.001$, and an inferential criterion conjoining P_1 and P_2 against an empirically derived threshold, each in its own test with its own composite verdict (§1.4.6); the engineering response to a fired observational verdict is investigation, not threshold debate.

The methodology's decision rules apply to every inferential claim about a population parameter, and the Wilson interval to every descriptive one. Observational criteria make no inferential claim and lie outside both by construction.

Optional contextual zero-failure bound. An observational verdict makes no population-level claim. For transparent reporting only, the framework MAY annotate a passing observational verdict with a contextual upper bound on the failure probability under an explicitly-stated i.i.d. Bernoulli model. The classical ”rule of three” (Hanley & Lippman-Hand, 1983; cross-referenced in §9) gives, under that model, an approximate 95% upper bound of $3/n_c$ on the failure probability when zero failures are observed in n_c trials. Where shown, the annotation carries the explicit i.i.d. Bernoulli caveat and does not change the verdict label, which remains observational. The methodology specifies the report wording as:

No failure of C_c was observed in n_c trials. This criterion is configured as observational, so the verdict itself makes no population-level claim. For context only, under an i.i.d. Bernoulli model, zero failures in n_c trials corresponds approximately to a 95% upper bound of $3/n_c$ on the failure probability.

High-consequence label override. For high-consequence safety criteria, the framework MAY (and the methodology recommends it) suppress the literal "PASS" label below an operator-configured minimum probe count and report **NO FAILURE OBSERVED** instead. The substitution is a label change only; the verdict semantics (observational, deterministic on the run) are unchanged.

1.4.5a The criterion denominator, and why a trial fails A criterion is evaluated on every trial of the sampling — every sample is presented to every criterion; there is no per-criterion filtering of inputs. On each trial the criterion yields exactly one of two outcomes:

- **PASS** — the criterion produced a value to test and its postcondition holds.
- **FAIL** — anything else. A FAIL carries a **reason**, which is diagnostic only and does not change the arithmetic:
  - condition — the postcondition was evaluated and did not hold;
  - transform / no-value — no value to test could be produced: the service returned malformed output, timed out, omitted required material, refused, or a transformation step failed, so the postcondition could not even be reached.

A criterion's denominator n_c is therefore simply the size N of the sampling; the success count K_c is the number that PASSed; $\hat{p}_c = K_c/n_c$. **Every trial counts.** A trial that produced no testable value is a FAIL like any other — its reason tells the developer it was not the postcondition that failed, but the trial is neither excluded from the denominator nor counted as a success. This is a single, uniform rule: there is no per-criterion denominator policy, no enum to choose, and no exclusion of unproducible trials.

Terminology. Earlier drafts labelled the transform/no-value case INCONCLUSIVE at the per-trial level. That was a misnomer: such a trial is not "inconclusive" — it is a FAIL whose reason is recorded. INCONCLUSIVE has its proper meaning only at the **verdict** level (§1.4.5, §1.4.6): a criterion's verdict is INCONCLUSIVE when the statistical procedure cannot render PASS or FAIL at all — no outcome could pass the compliance test at the observed size, $n_c = 0$, or covariates are misaligned. A per-trial transform failure is a FAIL; a criterion-level INCONCLUSIVE is a non-determination. The two must not be conflated.

Every sample is seen by every criterion, so nothing narrows a functional criterion's denominator below N. The one place a denominator legitimately narrows anywhere in the methodology is the latency dimension (§12.2.1): latency is summarised over the samples that pass every functional criterion, whereas a functional criterion's denominator is always the full sampling.

Availability as its own criterion. Because a non-response is already a FAIL of any criterion that needed the response, a quality criterion's rate is end-to-end — it reflects unavailability with no special handling. A contract that wants availability as a distinct, visible metric simply declares it as its own criterion (for example a `zeroFailures` or rate criterion over "the service returned usable output"). That is ordinary structural composition, not a denominator mechanism; there is no `denominatorPolicy` and no `availabilityCriterionRef` gate on the denominator.

The transparent-statistics output (§10.2) exposes, per criterion, n_c, K_c, $\hat{p}_c$, and a breakdown of the FAILs **by reason** — in particular how many were transform/no-value failures rather than condition failures. A high transform/no-value share is itself a diagnostic signal: a service producing testable output only intermittently is structurally unhealthy in a way that warrants attention before the pass rate is read (§1.4.7).

1.4.5b Conditional containment model for architectural discharge Architectural discharge — the guardrail pattern — leaves a statistical shadow, which this section models — taking care not to confuse the shadow with the obligation that casts it. The obligation is genuinely categorical, and it may be stated on the service's own contract like any clause, where it is evaluated observationally: an observed violation falsifies it, but no finite run proves a "never", and no estimate of the generator's own success rate can. When the service cannot realistically be made to produce the prohibited content often enough to establish its reliability, the application's answer is to interpose an architectural boundary — a guardrail, a deterministic filter, a hard schema gate, or a refusal classifier — between the generator and the consumer; where that boundary is itself probabilistic it carries its own, closely related contract: an ordinary rate-bounded clause on how reliably it blocks the prohibited class of content. The statistical question then becomes the residual rate at which prohibited content clears both stages of the composite of generator-then-boundary.

Write p_{harm} for the probability that the stochastic generator emits content violating the categorical clause, and $p_{\text{miss}|\text{harm}}$ for the conditional probability that the architectural boundary fails to block such content given that it was emitted. By the multiplication rule for the probability of a conjunction, the residual probability that prohibited content reaches the consumer is their product:

$$p_{\text{residual}} = p_{\text{harm}} \cdot p_{\text{miss}|\text{harm}}.$$

The decomposition separates two quantities with different evidential sources:

Quantity	Meaning	How it is evidenced
p_{harm}	The upstream generator's prevalence of harmful candidate outputs	Production monitoring, sentinel streams, representative production sampling, or not estimated

Quantity	Meaning	How it is evidenced
$p_{\text{miss}\|\text{harm}}$	The guardrail's conditional false-negative probability when presented with harmful content	Guardrail experiment over a harmful challenge sampling
p_{residual}	The end-to-end probability that harmful content reaches the user	Product of the two terms, if both are known or bounded under compatible population claims

Any production estimate of p_{harm} must observe the generator's pre-guardrail candidate output — directly, or through an equivalent shadow or sentinel stream that sees what the generator proposed before the guardrail suppressed or transformed it. Post-guardrail production monitoring sees only what reached the user, and therefore estimates p_{residual}, not the upstream prevalence p_{harm}. This is an easy mistake to make in practice, where many safety monitors are positioned downstream of the very guardrail whose containment they are meant to be independent of.

The architectural-discharge experiment primarily estimates or bounds the second term, $p_{\text{miss}|\text{harm}}$. It does not, by itself, estimate p_{harm}. A clean guardrail experiment therefore supports a statement of the form:

> Conditional on harmful content being presented to the guardrail, and relative to the specified challenge sampling or challenge distribution, the guardrail's miss probability is bounded above by the reported value at the configured confidence level.

It does not support, by itself, the statement:

> Harmful content reaches users in production at no more than this rate.

That stronger end-to-end claim requires a separate estimate or bound on the upstream generator harm prevalence p_{harm}, and the two quantities must be indexed to compatible populations.

Two distinct samplings. Evidence about this architecture is gathered over two samplings that are easily conflated and must be named apart, because they vary different things and estimate different quantities:

- A **red-team-input sampling** — adversarial inputs submitted to the generator-plus-guardrail application (the V_{probe} form of §1.4.8: prompts crafted to elicit prohibited content). Exercising it estimates the end-to-end behaviour of the composite under adversarial prompting, $P(\text{harm reaches user} \mid \text{adversarial prompt})$. It does not, by itself, isolate the guardrail's conditional sensitivity, because a clean run conflates "the generator rarely produced harmful candidates" with "the guardrail blocked the ones it did."

- A **harmful-candidate sampling** — prohibited candidate outputs presented directly to the guardrail (the V_{harm} form of the worked example below). Exercising it estimates the conditional containment term $P(\text{guardrail misses} \mid \text{harmful candidate presented})$, which is exactly $p_{\text{miss|harm}}$. It says nothing about how often the generator emits such candidates.

Both samplings are useful and both may evidence the same guardrail, each in its own test with its own sampling, but a sensitivity bound derived from the harmful-candidate sampling must not be reported as though it came from end-to-end adversarial probing, nor vice versa.

Guardrail sensitivity and miss probability. For a harmful challenge sampling of size n_H, let K_B be the number of harmful candidate outputs correctly blocked by the guardrail. Define the guardrail sensitivity estimate

$$\hat{s} = \frac{K_B}{n_H}.$$

The corresponding observed miss rate is

$$\hat{q} = 1 - \hat{s}.$$

An inferential guardrail-sensitivity criterion reports the one-sided Clopper–Pearson lower bound (§2.3.3)

$$L_s(\alpha)$$

on the true sensitivity s — the probability that the guardrail blocks a harmful candidate output given one is presented to it. The exact bound is used because containment is an affirmative safety claim, and the Wilson bound's coverage near the boundary can fall short of its nominal level (at zero observed misses its non-coverage tends to e^{-z^2}, about 6.7% at a nominal 5%). The corresponding upper confidence bound on the conditional miss probability is

$$U_{\text{miss}}(\alpha) = 1 - L_s(\alpha).$$

If the upstream prevalence p_{harm} were known, the residual end-to-end harm probability would be bounded as

$$p_{\text{residual}} \leq p_{\text{harm}} \cdot U_{\text{miss}}(\alpha),$$

provided the challenge sampling is admitted as representative of the harmful-output population to which the claim is being extended, and provided the production architecture admits no bypass around the guardrail. In practice p_{harm} is itself estimated, and the product of an estimate with a bound is not a bound. A joint statement needs

an upper confidence bound on each term, at levels that add up: with $U_{\text{harm}}(\alpha_1)$ an upper bound on the prevalence at level $1 - \alpha_1$,

$$p_{\text{residual}} \leq U_{\text{harm}}(\alpha_1) \cdot U_{\text{miss}}(\alpha_2) \quad \text{with confidence at least } 1 - \alpha_1 - \alpha_2$$

by the union bound, under the same representativeness and no-bypass provisos.

The experiment has bounded the containment term, $U_{\text{miss}}(\alpha)$, not the upstream prevalence term p_{harm}.

Multiple boundaries. Real architectures often interpose several boundaries in series — a prompt classifier, a retrieval filter, an output classifier, a schema gate, a deterministic deny-list, a human-escalation step. The containment term is then the conditional probability that all relevant boundaries miss given harm, $P(M_1 \cap \cdots \cap M_r \mid H)$. The individual miss probabilities must not be multiplied: that step assumes conditional independence, which logical composition does not supply (the same adversarial content that defeats one classifier is exactly the content likely to defeat another). Absent a declared and justified conditional-independence model, the term must be evaluated by the chain rule $P(M_1 \mid H)\, P(M_2 \mid H, M_1) \cdots$, or — more practically — estimated end-to-end over a harmful-candidate sampling presented to the full boundary stack.

Relationship to observational zero-failures criteria. The observational zero-failures criterion of §1.4.5 remains deterministic: it reports whether any miss was observed. It does not itself become an inferential criterion. The guardrail's own evidence is the inferential guardrail-sensitivity criterion of this section, which leads; the observational criterion complements it. The two typically run over different samplings, and so in separate tests: the observational criterion is naturally exercised over a red-team-input sampling (was any miss observed end-to-end under adversarial prompting?), while the inferential sensitivity criterion is exercised over a harmful-candidate sampling presented directly to the guardrail (what lower confidence bound is supported for the guardrail's sensitivity, and what upper bound follows for its miss probability?). When they are deliberately run over one sampling, they are two criteria of one test and that shared sampling must be stated; it cannot be assumed.

False positives. A guardrail that blocks every candidate output has perfect sensitivity but no practical utility. For this reason the guardrail's false-positive behaviour should be modelled separately over a non-harmful challenge sampling. Write p_{fp} for the false-positive probability — the probability that the guardrail blocks content given that the content is non-harmful — and $p_{\text{specificity}} = 1 - p_{\text{fp}}$ for the corresponding specificity.

The specificity criterion is then constructed symmetrically with the sensitivity criterion above, under the same exact construction. For a non-harmful challenge sampling of size n_N, let K_A be the number of non-harmful candidate outputs correctly allowed by the guardrail, giving the observed specificity

$$\hat{t} = \frac{K_A}{n_N}.$$

An inferential specificity criterion may report the one-sided Clopper–Pearson lower bound $L_t(\alpha)$ on the true specificity, and hence the upper confidence bound on the false-positive probability

$$U_{\text{fp}}(\alpha) = 1 - L_t(\alpha).$$

The sensitivity and specificity criteria are reported separately, over separately named harmful and non-harmful challenge samplings, so that over-refusal evidence is as inspectable as under-blocking evidence. Which rate a contract constrains, and how it guards the other, is contractual; Distributional Contracts §8.9 sets it out, following the Neyman–Pearson principle. The statistical consequence is this. A guardrail verdict that requires both criteria to pass is an **intersection–union test** (Berger, 1982): its probability of declaring the sensitivity requirement met when it is not is at most the sensitivity criterion's α, whatever the specificity criterion does, because a pass requires the sensitivity criterion to pass on its own. The specificity criterion therefore costs sample size, never the sensitivity claim's error rate, and no multiplicity correction is needed to protect that claim. The union-bound envelopes of §1.4.6 bound a different event, one or more criteria wrongly passed, and remain valid alongside.

Population-claim discipline. The challenge sampling's `populationClaim` governs the reach of the evidence. If the harmful probe set is a finite corpus, the result is a claim about that corpus. If the operator declares a superpopulation claim, the verdict must name the challenge distribution and record the representativeness argument. If the probes are exploratory, the result is no-generalisation evidence. In no case does an adversarial challenge result automatically become a production-traffic claim.

Stratification and the target estimand. A harmful challenge sampling is rarely homogeneous: it may span failure modes (self-harm, medical misinformation, jailbreaks, PII leakage, illegal-content requests), severity bands, languages, prompt styles, and template families. A pooled sensitivity bound can therefore look excellent overall while concealing weak containment in a rare but critical stratum. The general machinery for this is already in place (§8.2.1, §8.4.6): the verdict must declare its `targetEstimand` — call-weighted, prompt-weighted, stratum-weighted, severity-weighted, or no-generalisation — and where the harmful-candidate set is stratified by failure mode or severity the report should either give per-stratum sensitivity bounds or state the weighting scheme by which the pooled bound is to be read. A pooled bound must not be read as a guarantee for every harm stratum unless the per-stratum evidence supports that reading. For safety work this is not statistical fastidiousness: it is what stops a guardrail that is excellent in aggregate from quietly failing a small high-severity class.

The model therefore licenses only conditional claims — a bound on the guardrail's miss probability, and, given a separate estimate of p_{harm}, a bound on the residual harm rate under the stated population and no-bypass assumptions — never a claim that the categorical clause has been proved. In short: the guardrail experiment does not estimate how often the generator is dangerous; it estimates how much containment remains when it is.

Worked example: guardrail containment for clinical advice A clinical-advice service contains a categorical clause:

> The service shall not emit harmful medical advice.

Because the LLM cannot realistically be coaxed into producing such advice often enough to establish its reliability, a guardrail is interposed between the LLM and the user. The guardrail is tested separately over a harmful challenge sampling

$$V_{\text{harm}}$$

containing $n_H = 100{,}000$ pseudo-LLM responses classified as harmful by the project's safety taxonomy and attested independent from the guardrail's training data.

The guardrail blocks $K_B = 99{,}990$ of the harmful candidate outputs.

The observed sensitivity is

$$\hat{s} = \frac{99{,}990}{100{,}000} = 0.9999.$$

The calculation that follows requires V_{harm} to be declared a superpopulation sampling from the named harmful-candidate distribution (`populationClaim: superpopulation`); the bound is then a confidence statement about that distribution. If instead V_{harm} is an exhaustively evaluated finite corpus, the realised sensitivity on that exhaustive run is known exactly as $99{,}990/100{,}000$ — there is no uncertainty about that realised run, and a binomial confidence bound is not a confidence bound for it. The corpus's expected sensitivity, over repeated presentations of the same candidates, is known exactly from one run only if the guardrail is deterministic for a fixed candidate output — an assumption to be stated, not presumed; otherwise it is a claim that needs repeated presentation (§8.4.6). Any extension beyond the corpus is then a separate superpopulation claim, declared and argued as such (§8.4.6). The calculation below assumes the superpopulation declaration.

The calculation also treats the $n_H = 100{,}000$ harmful trials as independent Bernoulli observations under that declared estimand. This must be checked, not assumed: if the 100,000 candidates were generated from a much smaller number of prompt templates, model outputs, paraphrase families, or synthetic augmentations — as "pseudo-LLM responses" readily can be — the effective evidence is far below 100,000 and plain binomial calibration overstates it. Where the challenge set carries such material clustering, the criterion must use an estimator appropriate to the design (§8.2.1) or demote the claim per §8.4.6, exactly as for any other rate-bounded criterion.

Using the one-sided Clopper–Pearson lower bound at $\alpha = 0.05$ gives

$$L_s(0.05) \approx 0.999830.$$

The corresponding upper confidence bound on the conditional miss probability is

$$U_{\text{miss}}(0.05) = 1 - 0.999830 \approx 0.000170.$$

Therefore, relative to the stated harmful challenge distribution, the guardrail experiment supports the statement:

> At one-sided 95% confidence, the guardrail's conditional miss probability on harmful candidate outputs is bounded above by approximately 1.70×10^{-4}.

Equivalently, the guardrail misses at most approximately 1.70 harmful candidate outputs per 10,000 harmful candidate outputs, at the stated confidence level and under the stated modelling assumptions.

The residual end-to-end harm probability is

$$p_{\text{residual}} = p_{\text{harm}} \cdot p_{\text{miss|harm}}.$$

The experiment has bounded the second term. It has not estimated the first. Thus the end-to-end claim remains conditional on the prevalence:

$$p_{\text{residual}} \leq p_{\text{harm}} \cdot 0.000170 \quad \text{(for a known } p_{\text{harm}}\text{)}.$$

If, hypothetically, a separate pre-guardrail sentinel stream supported a one-sided upper bound on the prevalence at level 97.5%,

$$U_{\text{harm}}(0.025) = 0.001,$$

the joint statement pairs it with the guardrail bound at the same split level, $U_{\text{miss}}(0.025) \approx 0.000184$, and bounds the residual harm probability, with confidence at least 95%, by

$$0.001 \times 0.000184 = 1.84 \times 10^{-7}.$$

That is approximately 1.84 harmful responses reaching the user per 10 million production interactions, subject to both bounds being valid for compatible populations and subject to the production architecture admitting no bypass around the guardrail.

Without the separate bound on p_{harm}, the companion must not make the ”1.84 per 10 million” claim. It may only state the conditional containment result.

If the guardrail instead blocks all $100{,}000$ harmful challenge outputs, the observed sensitivity is

$$\hat{s} = 1.0.$$

This is not evidence of perfection. The one-sided Clopper–Pearson lower bound at $\alpha = 0.05$ is $0.05^{1/100{,}000}$, approximately

$$L_s(0.05) \approx 0.999970,$$

so the corresponding upper confidence bound on the conditional miss probability is

$$U_{\text{miss}}(0.05) \approx 3.00 \times 10^{-5}.$$

The correct statement is therefore:

> No misses were observed in 100,000 harmful challenge trials. Under the i.i.d. Bernoulli model, the guardrail's conditional miss probability is bounded above by approximately 3.00×10^{-5} at one-sided 95% confidence, relative to the specified harmful challenge distribution.

(The Wilson bound would give 2.71×10^{-5} here, understating the exact bound.)

The incorrect statement is:

> The guardrail is perfect.

The experiment supports a strong containment claim, not a zero-risk claim.

1.4.6 The composite verdict and its Type-I envelope A contract's verdict is a structured tuple over its per-criterion verdicts. The tuple is not collapsed to a single PASS/FAIL at the reporting layer; a flat boolean verdict surface would obscure per-criterion outcomes that the contract requires the consumer to see.

Structural composite. Let $V_c \in \{\text{PASS}, \text{FAIL}, \text{INCONCLUSIVE}\}$ denote the verdict of criterion c (§1.4.5). The contract's structural composite verdict is

$$V_{\text{contract}} = \begin{cases} \text{PASS} & \text{if } \forall\, c : V_c = \text{PASS} \\ \text{FAIL} & \text{if } \exists\, c : V_c = \text{FAIL} \\ \text{INCONCLUSIVE} & \text{otherwise.} \end{cases}$$

with the per-criterion verdicts retained in full on the composite artefact. When the test also asserts latency constraints, this functional composite is V_{rate}, and the test's verdict V_{test} composes it with the latency verdict by the same rule (§12.3.2). A service judged against both a requirement and its baseline carries two criteria over the same postconditions, a compliance criterion and a regression criterion (see Two Testing Paradigms); the rule composes them like any other criteria, and the compliance criterion's α enters the false-compliance envelope and the regression criterion's the false-degradation-signal envelope. A consumer reads the contract verdict and the supporting per-criterion verdicts in one place.

Enforcement. The functional dimension is enforced by default: V_{rate} enters V_{test} and can fail the test. A run-time switch can make it advisory (§12.6). Each criterion is then decided by its rule exactly as above and V_{rate} is reported beside V_{test}, but it does not enter V_{test} and cannot fail the test. The switch acts on the functional dimension as

a whole, never on one criterion, and a configuration refusal (§5.7.1) applies whatever the mode.

The structural composite is the methodology's representation of ”the contract is satisfied.” A baseline may carry an aggregate $\widehat{p}$ over the conjunction of all postconditions as a descriptive statistic — useful for dashboards and trend reporting — but the aggregate is not threshold-bearing: thresholds are not derived from it and verdicts are not framed against it. The hiding result of §1.4.4 demonstrates why a conjunction-based threshold is structurally unable to detect the per-criterion movement the methodology exists to surface.

Type-I envelopes by procedure direction. Per-criterion procedures of §3.2 control different error events, and a single unlabelled ”composite Type-I envelope” mixes them. The methodology therefore reports two procedure-specific envelopes, each a union-bound aggregate over its own family of inferential criteria:

- **False-compliance envelope** — the long-run probability, bounded by the sum of per-criterion α_c over **compliance** criteria, of declaring one or more compliance criteria satisfied when the corresponding true rate is at or below its requirement:

$$\alpha_{\text{false-compliance}} \leq \sum_{c \in \text{compliance}} \alpha_c.$$

- **False-degradation-signal envelope** — the long-run probability, bounded by the sum of per-criterion α_c over **regression** criteria, of issuing one or more degradation signals when no service has changed — each criterion's baseline and test drawn afresh under an unchanged success probability:

$$\alpha_{\text{false-degradation-signal}} \leq \sum_{c \in \text{regression}} \alpha_c.$$

A contract whose criteria are all of one kind reports the single corresponding envelope; a contract that mixes compliance and regression criteria reports both, labelled by direction. A combined unlabelled aggregate is ambiguous because the two envelopes describe different error events and may, in a given contract, be set at different magnitudes. When the test also asserts latency constraints (§12.3.1), its envelopes cover those decisions too: each explicit latency requirement decided under VERIFICATION adds its α_j to the false-compliance envelope, and each baseline-derived latency constraint adds its α_j to the false-degradation-signal envelope. The envelopes cover the enforced dimensions only: an advisory dimension (§12.6), functional or latency, is decided and reported, but none of its decisions can fail the test, so its criteria and constraints add nothing to either envelope.

Exact. Each bound holds under arbitrary dependence among the per-criterion test statistics in its family; it is the union bound applied to the per-criterion Type-I events of that direction. It rests on each per-criterion error probability being at most its α_c, which the exact decision rules of §3.4 and §3.6 guarantee (for regression, in the experiment in which the baseline and the test are both random). Observational criteria do not contribute to either envelope because their verdicts are deterministic

on observation; they carry no α. A criterion decided under SMOKE intent (§5.7.3), by `compliance/smoke-exact-binomial` or `latency/smoke-compliance-exact-binomial`, does not contribute either: its α bounds a false FAIL of a service that meets its requirement, which is neither envelope's error event, and its PASS claims no compliance. Its α is stated with the criterion, beside its p-value.

Engineering guardrail. The two directions set α_c for different reasons. For a **compliance** criterion, α_c bounds the probability of declaring compliance that does not hold, so it is set by the consequence of a false compliance claim: a safety-class compliance criterion at $\alpha = 0.001$ holds at that level because a false claim of safety demands it. For a **regression** criterion, α_c bounds the probability of a false degradation alert, so it is set by the cost of a false alarm. The risk of missing a real degradation is not controlled by α_c: it is governed by the power of the test — the degradation to be detected, the target power and the sample sizes (§5.3, §5.4.1) — and at a fixed sample size a smaller α_c makes it larger. A safety-class regression criterion therefore guards against missed degradation through its sizing, not through a small α_c. The envelope $\sum_c \alpha_c$ is reported on the composite verdict as a disclosed property; it is not a control target, and per-criterion α_c is not adjusted to control it. A uniform reduction $\alpha_c \mapsto \alpha_c/m$ would lower each criterion's power — most consequentially for a regression criterion whose purpose is to detect a real degradation.

Projects whose governance demands a uniform composite α may apply a Bonferroni reduction at the operator level. The methodology does not impose it.

The companion's §7.3 addresses **cross-test** family-wise inflation when multiple independent contracts are evaluated in a suite. The §1.4.6 envelope addresses **intra-test** family-wise inflation across the criteria of a single contract. The two cases differ in source and in treatment.

1.4.7 Inferential reach of the sampling An experiment or probabilistic test is bound to a specific contract; the only thing it varies is the **sampling** — the single list of N samples it posts to the service. That one sampling is shared by every criterion the contract declares, and each criterion evaluates all N samples separately (§1.4.2); there is no per-criterion sampling. The criteria's outcomes on a shared sample are not statistically independent of one another (§1.4.6).

A criterion's verdict or evidence is, primarily, a claim about the sampling its experiment or test runs over (including observational PASS verdicts, which are claims about the samples observed rather than about a population). The descriptive Wilson interval, the regression integer cutoff, the compliance count k_{min}, the observational PASS verdict — all are statements about the N samples in the experiment's sampling and the responses the service produced for them. Extending the claim to a different input distribution (production traffic that the service ultimately faces, a hypothetical superpopulation the sampling is taken to represent) requires further evidence; the methodology does not sanction the extension implicitly. The finite-corpus and superpopulation framings of §8.4.6 set out the two interpretive moves available.

Criteria within an experiment share the sampling and so share its inferential reach.

Every per-criterion verdict in the experiment speaks to the same N samples; differences between per-criterion verdicts arise from differences in postcondition definitions, thresholds and error levels, not from differences in the input distribution. Evidence about a different input distribution comes from a separate test, bound to its own contract, whose sampling is drawn from that distribution; each test names its own sampling. A view across such tests — every test of one service, say — is a matter of reporting: it lists each test's verdict beside its sampling, and it is not the verdict of one run.

Verdict surface. Every per-criterion verdict names the sampling its experiment ran over — identifier, version, any policy-redaction metadata — so the reader knows exactly which N samples produced the evidence. Extending the claim to anything other than that sampling is the reader's interpretive move, not the verdict's.

Clinical-advice illustration. Evidence over a representative production sampling and evidence over an adversarial probe sampling come from two tests, each bound to its own contract and each with one sampling. The probe test's verdict (”no failure observed under adversarial probing of the self-harm failure mode across 200 samples”) is a statement about that probe sampling; it does not extend to a claim about self-harm advice in production traffic. The latter claim, if needed, requires sentinel monitoring or a SMOKE-intent diagnostic on the production sampling (§1.4.5); the methodology does not infer it from the probe test's verdict.

Sample-budget consequence. E tests over different input populations have E samplings, each of size N_e. Criteria within a test share N_e as their trial count (§1.4.5a); the limiting criterion within a test determines the test's required N_e. The total sample budget across the tests is $\sum_e N_e$; the limiting criterion is identified test by test, not by an aggregate sample count.

1.4.8 Worked example: the consult-advice tests The clinical-advice service has been threaded through this chapter; it is collected here as a single inspection point. Its requirements are evidenced over three input populations, so they are exercised by three tests, each bound to its own contract and each running over one sampling that every criterion of that contract shares: a structural test over production prompts, a readability test over inputs chosen to elicit clinical terminology, and an adversarial test over self-harm probes. This is the one canonical consult-advice example: §1.5.6 shows its baselines and §10.3 its transparent-statistics report, and every number is generated by the mavai-R oracle.

Postconditions. Four named predicates over the produced response.

- P_1: response parses as JSON
- P_2: required fields are present
- P_3: response is layperson-readable
- P_4: advice does not suggest self-harm

Samplings.

- V_{prod}: production prompts, redacted per policy XYZ, version 5. Version 5 names

the sampling specification — the production-prompt distribution and its redaction — and each run draws its prompts independently from it, so baseline and test hold different prompts from the same population. Declared claim: superpopulation (§8.4.6).
- $V_{\text{complexity}}$: inputs likely to elicit responses involving clinical terms a layperson would not recognise; version 2 names a curated generator of such inputs, from which each run draws independently. Declared claim: superpopulation, over the distribution the generator defines — which is what lets the exact compliance test of the readability contract speak about the service's readability on such inputs. Were $V_{\text{complexity}}$ instead a fixed curated list invoked once, the same count would support only a realised-run claim (§8.4.6).
- V_{probe}: a curated set of adversarial inputs designed to elicit self-harm advice; each example classified by failure mode within the `mavai-safety-taxonomy` v2; attested independent from the consult-advice guardrail's training data; version 3. Declared claim: finite corpus, the default for a probe set (§8.4.6); its criterion is observational and makes no population claim.

Contracts and criteria.

- `consult-advice@5`, tested over V_{prod}. $C_{\text{well-formed}}$: inferential, references $\{P_1, P_2\}$, empirical-origin threshold against baseline `consult-advice@2026-04-01` (951 successes in 1000 trials, §1.5.6), $\alpha = 0.05$, class `CORRECTNESS`. Its denominator is the full sampling (§1.4.5a); a malformed or absent response is a FAIL (reason: transform/no-value), counted like any other.
- `consult-advice-readability@1`, tested over $V_{\text{complexity}}$. $C_{\text{layperson-readable}}$: inferential, references $\{P_3\}$, contractual-origin threshold $p^*_{P_3} = 0.98$ at SLO origin, $\alpha = 0.001$, class `SAFETY`. Its denominator is the full sampling (§1.4.5a); a malformed or absent response is a FAIL (reason: transform/no-value), not an exclusion.
- `consult-advice-self-harm-probe@1`, tested over V_{probe}. $C_{\text{no-self-harm}}$: observational (`zeroFailures`), references $\{P_4\}$, no threshold parameter, class `SAFETY`. As an observational criterion it carries no rate denominator (§1.4.5a): one observed self-harm response fails it.

If the structural contract additionally wants availability as a distinct, visible metric, it declares a second criterion:

- $C_{\text{evaluable-response}}$: a criterion (`zeroFailures` or rate) over a response-evaluability postcondition, in `consult-advice@5` over V_{prod}. Like every criterion its denominator is the full sampling; it stands alongside $C_{\text{well-formed}}$ and does **not** alter its denominator.

The three runs. The structural test runs 1000 samples of V_{prod}, the readability test 800 of $V_{\text{complexity}}$ and the adversarial test 200 of V_{probe}. They observe:

- $\hat{p}_{C_{\text{well-formed}}} = 0.953$ over 1000 trials.
- $\hat{p}_{C_{\text{layperson-readable}}} = 0.985$ over 800 trials.
- $K_{C_{\text{no-self-harm}}} = 200$ — zero observed self-harm responses in 200 probes.

The decision artefacts, at per-criterion α:

- $C_{\text{well-formed}}$: the Fisher cutoff from the baseline's 951 of 1000, for a test of 1000 at $\alpha = 0.05$, is $c = 933$ (§3.4).
- $C_{\text{layperson-readable}}$: the exact test's smallest passing count for 800 trials against $p_{\text{req}} = 0.98$ at $\alpha = 0.001$ is $k_{\min} = 796$ (§3.6).

For description only, the one-sided Wilson lower bounds are 0.9407 ($C_{\text{well-formed}}$, $\alpha = 0.05$) and 0.9649 ($C_{\text{layperson-readable}}$, $\alpha = 0.001$).

Verdicts:

- `consult-advice@5`: PASS — 953 successes meet the cutoff of 933 derived from the baseline (displayed rate 0.933). Its false-degradation-signal envelope is $\alpha = 0.05$.
- `consult-advice-readability@1`: FAIL — 788 successes fall short of $k_{\min} = 796$; compliance with the 0.98 requirement is not demonstrated, although the observed rate, 0.985, is above it. Its false-compliance envelope is $\alpha = 0.001$.
- `consult-advice-self-harm-probe@1`: PASS — observational, zero failures observed across V_{probe}. The verdict carries no inferential claim; the statement made is exactly ”no failure of $C_{\text{no-self-harm}}$ was observed across 200 probe trials drawn from V_{probe} v3.” It enters neither envelope.

Each verdict is the composite of its own run (§1.4.6), over its own sampling. A report across the three tests — the service's standing on the consult-advice requirements — lists the three verdicts, each beside its sampling, its envelope and, for the empirical criterion, its baseline reference; it is a reporting view, not a verdict of one run, and it does not pool the three samplings. The reader of the report — operator, auditor, regulator — sees each test's conclusion, the supporting per-criterion evidence, the population each piece of evidence speaks to, and the disclosed false-alarm budget under which each conclusion was issued.

Under the clause-type taxonomy (see Clause Forms: Rate-Bounded and Categorical), $C_{\text{no-self-harm}}$ in this example is the observational zero-failures criterion evaluating the consult-advice service's responses to adversarial input. The associated categorical clause — the consult-advice service shall not emit self-harm advice — is evaluated by $C_{\text{no-self-harm}}$, which can falsify it but cannot establish it; the architectural commitment that interposes the guardrail between the model and the user is what bounds the residual risk. The probe test runs the service end to end, generator and guardrail together, so its clean result is evidence about that composite under adversarial input: it could mean that the generator produced no harmful candidate or that the guardrail blocked those it did produce, and it does not measure the guardrail's sensitivity on its own. That sensitivity is measured by a separate test whose sampling is a set of harmful candidates presented to the guardrail directly (§1.4.5b). A report reader who sees the probe test PASS is reading evidence that the service held over 200 probes, not evidence that the model would not have erred without the guardrail. How the commitment is declared and how V_{probe}'s coverage is recorded are contractual matters, set out in Distributional Contracts §8.9; the guardrail's false-positive rate enters as its own rate-bounded clause (Clause Forms).

1.4.9 Per-criterion trials in subsequent chapters Each per-criterion trial is a complete statistical object — independent of the others as a unit of modelling, though not necessarily statistically independent of them. The remainder of the companion treats per-criterion trials transparently: where §§2–12 develop estimation, cutoff derivation, sample sizing, the feasibility gate, transparent-statistics reporting, and latency, the single-criterion X_i, $\hat{p}$, α, and decision artefacts they discuss apply equally to each per-criterion trial $X_{i,c}$, $\hat{p}_c$, α_c, and p_c^*. No section that follows needs to be re-derived for the per-criterion case.

The composite verdict over the per-criterion verdicts (§1.4.6) and the procedure-direction-specific Type-I envelopes of the composite (§1.4.6) are the constructs particular to the partition; they have no single-criterion analogue.

The following chapter develops one further statistical consequence:

- **Per-criterion baselines** — how a measurement run emits a baseline vectorised across criteria, and how cutoffs are derived at resolution time rather than at emission.

1.5 Baselines

1.5.1 The baseline as the link between measurement and inference For an empirical-origin criterion (§7.4) the decision threshold is a function of a prior observation rather than a contractual constant. The methodology resolves this in two phases. The **measurement phase** yields an estimator $\hat{p}$ of the population rate p over n trials of the contract. The **inference phase** derives the integer cutoff from the baseline's counts, the test's size and a stated level α (§3.4), and resolves the service's current behaviour against it. The **baseline** is the statistical object that links the two phases: the estimator together with the conditions under which it was measured, which identify the population it estimates. Section 1.5.2 gives the formal definition, generalised per criterion.

1.5.2 The baseline as a family of per-criterion estimators A baseline is a family of per-criterion point estimators $\{\hat{p}_c\}$, one per criterion c declared on the contract. The family's structure follows §1.4.3: for each criterion c, the baseline stores:

- n_c, the number of trials for c, equal to the sampling size N (§1.4.5a);
- K_c, the number of those trials that PASSed;
- $\hat{p}_c = K_c/n_c$ for inferential criteria; the same pair feeds the deterministic rule for observational criteria (§1.4.5);
- optionally, for diagnostics, the split of the $n_c - K_c$ fails by reason (condition vs transform/no-value) — equivalently $n_{c,\text{evaluable}}$, the count that produced a testable value, and $r_{c,\text{obs}} = n_{c,\text{evaluable}}/n_c$.

The structural reference under which n_c is defined — the postconditions and the criterion — is part of the criterion's meaning. A baseline and a test whose matching

criteria differ in it differ structurally; under VERIFICATION the mismatch is a configuration error (§8.4.5).

The index of a baseline. The family $\{\hat{p}_c\}$ is interpretable as an estimator of the per-criterion population rates $\{p_c\}$ only relative to the conditions under which it was measured. A baseline is **indexed** by the pair (service-contract identity, resolved covariate values), and this index is its selection key: a test computes its own index and resolves against the baseline whose index matches (§1.5.3).

- The **service-contract identity** — in practice an ID or class name — names the contract whose criteria the baseline observes, and carries the **structural reference** (the postcondition-and-criterion structure under which the per-criterion trials are defined, fixing the meaning of each c in the family). A baseline and a test under different structural references are different contracts; the methodology treats that mismatch as a structural error (§8.4.5), not as an index divergence to be adjudicated. The service-contract identity is not a covariate.
- The **resolved covariate profile** (§8.4.1) — the operating regime of the measurement run, resolved to concrete values. §1.5.3 develops its role.

Two further things qualify a baseline without being part of its index:

- the **factor record** (§1.3.1) — service deployment, model, temperature, serving configuration — is provenance: a record of the system that was measured. It does not by itself drive selection. A project may — and in most cases should — declare the factor dimensions that matter (model ID, serving stack, and so on) **as covariates**, at which point their resolved values enter the index through the covariate channel; but the programming model does not force the factor record and the covariate profile to coincide.
- an optional **expiration window** (§8.4.2) — where declared, the temporal scope beyond which the baseline is no longer admitted as a reference, bounding the stationarity assumption of §1.3 in time (§1.5.5).

Each of these — index components and qualifiers alike — is a property of the baseline as a whole, not of an individual $\hat{p}_c$ within it.

The per-criterion cutoff c_c and level α_c are **not** part of the baseline. They are properties of the inference that consumes the baseline, not of the observation that supports it; §1.5.4 develops the consequence.

1.5.3 Covariates and the baseline's interpretive scope The role of the covariate profile in the baseline is more consequential than any single component of its structure. It is the covariate profile that makes the baseline's observations interpretable as a reference for future tests.

Each $\hat{p}_c$ is conditional on the covariate profile. The working approximation of §1.3 — that trials are i.i.d. samples from a Bernoulli distribution with parameter p_c — holds at the population level under the contextual conditions of the measurement run. A measurement taken on a weekday morning in the EU under a specific deployment configuration produces a $\hat{p}_c$ that is an estimator of p_c under those conditions. The

same service measured on a weekend evening in a different region under serving-stack pressure may yield a different population p_c. The covariate profile is what the baseline records to make the condition explicit, so the consuming test can ask whether its own context is comparable.

Comparability between baseline and test depends on covariate matching. A project may hold more than one baseline for a contract — for instance one per operating regime, distinguished by their index (for a fixed contract, by their resolved covariate values). A test resolves against the baseline whose index matches its own: where the test's covariate profile matches a baseline's, that baseline's $\widehat{p}_c$ is admitted as a reference for the test's inference. Where no matching baseline is available, the methodology treats the divergence as diagnostic: the test may proceed under explicit acknowledgement that the nearest baseline was measured under different conditions, or it may decline to resolve empirically at all. The mechanism by which a project decides between these options is governed by the project's covariate policy (§8.4.1) and is operational rather than methodological; the methodology's contribution is to require that the appropriate baseline is selected by matching index, and that any inference made under a divergent index carries that divergence on the verdict.

The covariate profile applies to the baseline as a whole, not per criterion. The covariates the project declares — time of day, day of week, deployment region, serving-stack version below the granularity the factor record pins, and so on — are properties of the measurement run that affect every criterion's observations simultaneously. They are recorded once for the baseline, not once per criterion. A reader inspecting the baseline reads one covariate profile, regardless of how many criteria the baseline carries observations for.

The covariate profile is what makes the baseline an audit artefact. Without it, the baseline is a vector of $\widehat{p}_c$ values with no anchor; a regulator inspecting the baseline cannot tell whether the conditions under which the measurement was taken match the conditions under which the service operates in production. With it, the baseline carries the contextual identity of the measurement and is inspectable end to end: this is the factor record of what was measured, under these covariate conditions, with these per-criterion outcomes, valid until this expiration. The baseline's role in Appendix A — as the empirical-evidence layer of the methodology's artefact stack — rests on the covariate profile being a first-class component of the artefact.

The methodology does not require any specific set of covariates; it requires only that whatever covariates the project declares are recorded on every baseline and on every test that consumes a baseline. The §8.4.1 machinery defines the per-covariate matching discipline. This chapter elevates that machinery to a defining role in the baseline artefact.

1.5.4 Threshold derivation at resolution time The baseline is an estimator. The cutoff against which a test compares the service's behaviour is derived at the moment the test resolves against the baseline, not at the moment the baseline is produced.

For an inferential criterion with origin EMPIRICAL, the integer cutoff c_c is derived

by the one-sided Fisher exact test of §3.4 from the **baseline's** success count K_c^{baseline} and trial count n_c^{baseline}, the **test's** sample size $n_{c,\text{test}}$, and the test's α_c:

$$c_c \;=\; \text{FisherCutoff}\big(K_c^{\text{baseline}},\; n_c^{\text{baseline}},\; n_{c,\text{test}},\; \alpha_c\big).$$

Both samples enter the calculation. The baseline's counts carry its finite-sample uncertainty — a baseline of 951 of 1000 and one of 95 of 100 have the same rate but very different evidential weight, and the rule treats them differently — and the test's size carries the test's. The false-degradation-signal probability the cutoff controls is computed with both counts random (§3.4); no sample-size ratio substitutes for the calculation, and the baseline is not assumed infinite. An empirical test larger than its baseline is refused (`TEST_LARGER_THAN_BASELINE`, §5.7.1).

Larger measure experiments produce more stable references and usually support more sensitive regression tests, and the methodology recommends that a measure experiment be materially larger than the regression tests that consume it, where cost permits. That is guidance about precision and power, not the basis of the calibration.

The deferral of cutoff derivation to resolution time is what allows the same baseline to support more than one test that disagrees on α_c or on $n_{c,\text{test}}$. The baseline is an observation; the test is the inference; the inference's parameters belong to the test that draws it.

Engineering guardrail. A baseline consumed by a test contributes, to the test's verdict, the baseline's identifier and the per-criterion counts K_c^{baseline} and n_c^{baseline}. The verdict also records the test's $n_{c,\text{test}}$ and α_c, the decision rule and its version, and the resulting c_c and $c_c/n_{c,\text{test}}$. An auditor reading the verdict has the full set of inputs to the Fisher calculation and can reproduce the cutoff.

For an inferential criterion with origin SLA, SLO, or POLICY, the threshold is contractual and the baseline does not enter the decision; the baseline's $\widehat{p}_c$ may still be carried on the verdict as diagnostic context, but it does not parameterise the test. For an observational criterion there is no threshold; the baseline's per-criterion observation is carried as diagnostic context but is not consulted at verdict time.

1.5.5 Expiration A baseline may optionally carry an expiration window bounding its temporal scope. Where one is declared — per §8.4.2 — it bounds the baseline's validity (§1.5.2) to a half-open interval, beyond which the baseline is no longer admitted as a reference for inference. Expiration, when present, is at the baseline level rather than per-criterion: the temporal scope is a property of the operating regime the covariate profile encodes — time of day, day of week, deployment region, and the like — which conditions every $\widehat{p}_c$ simultaneously. A baseline so bounded is stale uniformly; while current it supports every criterion's resolution at once. A baseline with no expiration window carries no temporal bound and is admitted until superseded by project or operational policy.

1.5.6 Worked example The baselines of the one canonical consult-advice example (§1.4.8, §10.3): one measurement run per contract of §1.4.8, each over one sampling that every criterion of the contract shares, so every n_c in a baseline equals that run's sampling size N.

Measurement conditions, common to the three runs.

- Factor record: `consult-advice-service@3.1`, model `claude-sonnet-4-5-20250929`, temperature `0.0`, system-prompt `consult-advice-prompt@5`.
- Covariate profile: `day_of_week = WEEKDAY`, `time_of_day = 08:00–12:00`, `region = EU`, `serving_stack = standard`.
- Expiration: `2026-08-13` (90 days from emission).

Per-criterion observations, one baseline per structural reference.

Structural reference	Sampling	N	Criterion	K_c	$\hat{p}_c$
`consult-advice@5`	V_{prod} v5	1000	`well-formed`	951	0.951
`consult-advice-readability@1`	$V_{\text{complexity}}$ v2	800	`layperson-readable`	788	0.985
`consult-advice-self-harm-probe@1`	V_{probe} v3	200	`no-self-harm`	200	1.000

Tests consuming these baselines. A subsequent test of each contract under matching factor record and matching covariate profile resolves its per-criterion decision artefacts from that contract's baseline:

- $C_{\text{well-formed}}$ (origin EMPIRICAL, $\alpha = 0.05$, test sample size $n_{\text{well-formed,test}} = 1000$): the Fisher cutoff from the baseline's 951 of 1000 is $c = 933$ (§3.4). The cutoff is recorded on the verdict together with the baseline's counts and identifier.
- $C_{\text{layperson-readable}}$ (origin SLO, $p_{\text{req}} = 0.98$, $\alpha = 0.001$): the threshold is contractual; the baseline does not parameterise it. The baseline's $\hat{p}_{\text{layperson-readable}} = 0.985$ is recorded as diagnostic context.
- $C_{\text{no-self-harm}}$ (observational): no threshold is derived; the test evaluates whether any failure of the criterion is observed in its own run. The baseline's observation is recorded as diagnostic context on the verdict.

A test under a non-matching covariate profile — say, region = US — declines to resolve against the baseline without explicit project-policy acknowledgement of the divergence. The verdict reports the divergence; the auditor reads it.

1.5.7 Baselines in subsequent chapters The baseline is the statistical object that links measurement to inference: a family of per-criterion estimators, qualified by the factor record, covariate profile, expiration window, and structural reference under

which it was measured. Each baseline is recorded as a row of Appendix A's artefact stack.

The remainder of the companion treats baselines transparently. The existing §2 (Baseline Estimation) develops the single-criterion point-estimator and standard-error machinery, which applies per criterion under §1.4.3. §3 (Threshold Derivation) develops the Fisher cutoff that §1.5.4 invokes at resolution time. §4 (The Perfect Baseline Problem) treats the $\hat{p}_c = 1$ case, which is the common situation for observational criteria where every evaluable trial was a success, and which the Fisher cutoff handles without a special case. The existing §§8.4.1–8.4.2 develop the covariate and expiration machinery that §§1.5.3 and §1.5.5 condition on. No section that follows needs to be re-derived for the per-criterion case.

---

2. Baseline Estimation (Experiment Phase)

2.1 Point Estimation

Given n experimental trials with k successes, the maximum likelihood estimator (MLE) for p is:

$$\hat{p} = \frac{k}{n}$$

Example: In our JSON generation service contract, an experiment with n = 1000 trials yields k = 951 successes.

$$\hat{p} = \frac{951}{1000} = 0.951$$

2.2 Standard Error

The standard error of $\hat{p}$ quantifies the precision of the estimate:

$$\text{SE}(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Example:

$$\text{SE} = \sqrt{\frac{0.951 \times 0.049}{1000}} = \sqrt{0.0000466} \approx 0.00683$$

2.3 Confidence Intervals

The mavai methodology uses the Wilson score interval as its descriptive confidence interval for a single observed proportion. It describes the uncertainty around an observed rate; it decides no verdict — the decision rules are the exact procedures

of §3.4, §3.6 and §12.4. This section documents the Wilson method and, for completeness, includes the Wald (normal approximation) method that statisticians may encounter in textbooks.

2.3.1 Wilson Score Interval (the mavai Method) Wilson score intervals are the methodology's default descriptive interval family because they are stable near the boundary, remain inside $[0, 1]$, are simple to reproduce across languages, and generally have **much better coverage behaviour** than Wald intervals in the regimes encountered in probabilistic software testing. Wilson intervals are **not exact finite-sample intervals**; they are a score-interval construction selected for operational stability and good practical coverage. Where an affirmative claim needs exact calibration — a compliance verdict, a guardrail's containment bound — the methodology uses the exact binomial test and the Clopper–Pearson bound instead (§3.6, §1.4.5b); Wilson's non-coverage can exceed its nominal level near the boundary, tending to e^{-z^2} (about 6.7% at a nominal 5%) at zero observed failures.

In summary, Wilson:

- Has substantially better coverage than Wald across small n and proportions near 0 or 1, while remaining a score-test approximation rather than an exact procedure.
- Remains well-defined for proportions at or near the boundary (including $\hat{p} = 1$), where Wald collapses.
- Never produces bounds outside $[0, 1]$.

The Wilson interval endpoints are:

$$\frac{\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

Example (95% CI for $\hat{p} = 0.951$, $n = 1000$):

$$\text{Lower} = \frac{0.951 + \frac{1.96^2}{2000} - 1.96\sqrt{\frac{0.951 \times 0.049}{1000} + \frac{1.96^2}{4000000}}}{1 + \frac{1.96^2}{1000}} \approx 0.936$$

$$\text{Upper} \approx 0.963$$

Why Wilson as the Descriptive Interval? The mavai methodology uses Wilson for descriptive intervals because:

1. **Wilson is rarely worse than Wald in practice**: For large samples and moderate proportions, Wilson produces results nearly identical to the Wald approximation, so there is no operational penalty for using Wilson universally. (This is a comparative statement against Wald, not a claim of exactness against the binomial. See Brown, Cai & DasGupta, 2001; Newcombe, 1998.)

2. **Wilson avoids the worst Wald pathologies**: For small samples, extreme proportions, or perfect baselines ($\hat{p} = 1$), Wald produces degenerate or nonsensical results (zero-width intervals at the boundary, intervals outside $[0, 1]$). Wilson remains well-defined in all these cases. It is not exact — the operative claim is much-improved coverage behaviour, not perfect coverage.
3. **Consistency**: A single method ensures reported intervals are always comparable across tests. No verdict depends on the interval method.
4. **LLM testing reality**: High success rates ($p > 0.85$) are common in probabilistic testing of LLM-based systems. This is precisely where Wilson provides the most benefit.

What This Means for Developers

- Developers do not need to choose a method or configure thresholds for method switching
- Statistical calculations are consistent across all sample sizes
- The formulas shown above are what every mavai framework uses for descriptive intervals—always

Conformance Verification The mavai-R project generates reference Wilson score interval values using R's `qnorm` function. Each framework implementation verifies its Wilson computation matches the R-generated reference data. See `inst/cases/wilson_ci.json` and `inst/cases/wilson_lower.json` in the mavai-R repository.

2.3.2 Wald Interval (For Completeness) For completeness, this section documents the Wald interval (normal approximation), which statisticians will encounter in many textbooks. **The mavai methodology does not use this method**, but understanding it helps explain why Wilson is preferred.

For large n, by the Central Limit Theorem:

$$\hat{p} \overset{a}{\sim} N\left(p, \frac{p(1-p)}{n}\right)$$

The $(1 - \alpha)$ Wald confidence interval is:

$$\hat{p} \pm z_{\alpha/2} \cdot \text{SE}(\hat{p})$$

where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution.

Example (95% CI, $z_{0.025} = 1.96$):

$$0.951 \pm 1.96 \times 0.00683 = [0.938, 0.964]$$

Why the mavai Methodology Does Not Use Wald Statistics textbooks often present guidelines for when Wald is acceptable:

Condition	Textbook Guidance
$n \geq 40$ and $0.1 \leq \hat{p} \leq 0.9$	Wald acceptable
$n \geq 20$ and ($\hat{p} < 0.1$ or $\hat{p} > 0.9$)	Wilson preferred
$n < 20$	Wilson strongly recommended
$n < 10$	Wilson required; Wald inappropriate

Rather than implement conditional method selection, the methodology uses Wilson for every descriptive interval. This simplifies implementation while giving well-defined results in all cases—including the edge cases where Wald fails.

2.3.3 Clopper–Pearson Interval (Exact) The Clopper–Pearson method is an "exact" frequentist method for calculating a confidence interval for an unknown binomial proportion, such as a success or event rate observed over a fixed number of trials. It obtains the lower and upper limits by inverting binomial tests, rather than using a normal approximation, so it remains valid with small samples or proportions near 0 or 1. Its guaranteed coverage is at least the stated confidence level, which can make the resulting interval conservative.

The methodology uses it where a claim is affirmative or safety-class and must hold at the configured level exactly: its one-sided lower bound exceeds p_{req} exactly when the exact one-sided binomial test of §3.6 passes, so it is the interval form of the compliance decision, and it gives the guardrail bounds of §1.4.5b. Wilson (§2.3.1) remains the descriptive interval everywhere else.

2.4 Sample Size Determination

> **Epistemic status**: planning approximation based on normal asymptotics. Adequate for budgeting the precision of an estimate; not of the same epistemic type as the exact decision rules elsewhere in this document, and never used to size a test against them (§5.4, §5.5).

To achieve a desired margin of error e with confidence $(1 - \alpha)$, the required sample size is approximately:

$$n = \frac{z_{\alpha/2}^2 \cdot \hat{p}(1 - \hat{p})}{e^2}$$

Example: To estimate p ≈ 0.95 with ±2% margin at 95% confidence:

$$n = \frac{1.96^2 \times 0.95 \times 0.05}{0.02^2} = \frac{0.1825}{0.0004} \approx 456$$

Target Precision (95% CI)	Required n (for p ≈ 0.95)
±5%	73
±3%	203
±2%	456
±1%	1,825

2.5 Two Binomial Counts Given Their Total

A regression test compares two samples of the same binomial model: the baseline's $K_b \sim \text{Bin}(n_b, p_b)$ and the test's $K_t \sim \text{Bin}(n_t, p_t)$, independent. When the two success probabilities are equal, $p_b = p_t = p$, the conditional law of the test's count given the two counts' combined total $s = K_b + K_t$ does not depend on p:

$$P(K_t = x \mid K_b + K_t = s) \;=\; \frac{\binom{s}{x}\binom{n_b+n_t-s}{n_t-x}}{\binom{n_b+n_t}{n_t}},$$

the hypergeometric distribution — the probability that x of the s pooled successes fall among the test's n_t of the $n_b + n_t$ trials. It follows from the binomial model alone: the factors $p^s(1-p)^{n_b+n_t-s}$ cancel between the joint probability and the probability of the total. Its lower tail is the one-sided Fisher p-value on which the regression rule of §3.4 rests; no estimate of p is needed, because none enters.

Epistemic status. Exact result under the binomial model.

---

3. Threshold Derivation for Regression Testing

3.1 The Problem

The experiment established $\hat{p}_{\text{baseline}} = 0.951$ from $n_{\text{baseline}} = 1000$ samples. For cost reasons, regression tests will use $n_{\text{test}} = 100$ samples.

Question: What threshold p^* should the regression test use?

Naive approach: Use $p^* = 0.951$.

Problem with naive approach: With only 100 samples, the standard error is:

$$\text{SE}_{\text{test}} = \sqrt{\frac{0.951 \times 0.049}{100}} \approx 0.0216$$

Even if the true p equals the experimental rate, observed rates will vary. At $\pm 2\sigma$, we'd expect observations between 0.908 and 0.994. Using 0.951 as the threshold would cause frequent false positives.

The baseline itself is a sample too: 0.951 is an estimate from 1000 trials, not the true rate. A threshold that treats it as known ignores half of the uncertainty in the

comparison, and the regression rule of §3.4 therefore uses the baseline's counts, not only its rate.

3.2 One-Sided Hypothesis Testing Framework

Compliance and regression testing both use a one-sided binomial decision skeleton, but as introduced in Two Testing Paradigms they implement **two distinct hypothesis families with distinct error semantics**. Each is decided by its own exact rule, and the controlled error and the meaning of PASS/FAIL differ.

Compliance / assurance procedure (`compliance/exact-binomial`, §3.6) — affirmative test that the system meets a normative threshold p_{req}:

$$H_0 : p \leq p_{\text{req}} \qquad H_1 : p > p_{\text{req}}$$

Decision: PASS iff $K \geq k_{\min}$, the smallest count at which the exact one-sided binomial test rejects H_0. The procedure controls the **false-compliance** probability — the probability of declaring compliance when the true success probability is at or below the requirement — at or below α, exactly.

Regression / monitoring procedure (`regression/fisher`, §3.4) — test that detects degradation of the test's success probability p_t from the baseline's p_b:

$$H_0 : p_t \geq p_b \qquad H_1 : p_t < p_b$$

Decision: PASS if the observed test count $K_t \geq c$; FAIL if $K_t < c$, where the integer cutoff c is derived from the baseline's (K_b, n_b), the test size n_t and α. The procedure controls the **false-degradation-signal** probability — computed with the baseline count and the test count both random under an unchanged success probability — at or below α. A regression PASS means ”no degradation signal at this cutoff”; it does not establish equivalence to the baseline.

The displayed rate c/n_t accompanies the binding integer cutoff c. Implementations report both c and c/n_t; the integer cutoff is the decision artefact (§3.4, §5.1).

The differences across the two paradigms are summarised below. Note that the controlled error and the verdict semantics differ.

Paradigm	Threshold	Hypotheses	Error controlled	PASS means
Compliance	p_{req} (given)	$H_0 : p \leq p_{\text{req}}$ vs $H_1 : p > p_{\text{req}}$; PASS iff $K \geq k_{\min}$	False-compliance probability, at most α at every n	Evidence supports compliance at the configured level

Paradigm	Threshold	Hypotheses	Error controlled	PASS means
Regression	Derived from the baseline's (K_b, n_b)	$H_0 : p_t \geq p_b$ vs $H_1 : p_t < p_b$; PASS iff $K_t \geq c$	False-degradation-signal probability, baseline and test both random, at most α	No degradation signal at the configured cutoff

We seek decision rules that:

- control the stated error at or below α under the working model, with the error event named (because the binomial is discrete, the achieved error probability is generally below α; §3.4 and §3.6 state it);
- detect true violations or degradation with as much power as that control allows, stating the power exactly (§5.3).

3.3 Normal-Approximation Intuition for One-Sided Bounds

The formula in this subsection is shown only to motivate the idea of a one-sided lower bound. The methodology does not decide with it: regression verdicts use the exact rule of §3.4 and compliance verdicts the exact rule of §3.6.

The $(1 - \alpha)$ one-sided lower confidence bound is:

$$p_{\text{lower}} = \hat{p} - z_\alpha \cdot \text{SE}$$

Note: For one-sided bounds, we use z_α (not $z_{\alpha/2}$). A one-sided confidence level of $1 - \alpha$ is a property of the bound in repeated sampling; it is not, by itself, the false-positive rate of any test that uses the bound.

Confidence Level	z_α	One-sided level α
90%	1.282	0.10
95%	1.645	0.05
99%	2.326	0.01

3.4 Regression Cutoff (One-Sided Fisher Exact Test)

Empirical regression is decided by `regression/fisher`, version 1: the one-sided Fisher exact test, expressed as an integer cutoff on the test's success count. It uses the baseline's counts and the test's size jointly; no developer configuration beyond α is added.

Inputs. The baseline's success count K_b and trial count n_b, the test size n_t, and α.

Algorithm. For each possible test count $k_t = 0, \dots, n_t$, pool the successes, $s = K_b + k_t$, and compute the one-sided Fisher p-value — the hypergeometric lower tail, the probability that at most k_t of the s pooled successes fall among the test's n_t of the $n_b + n_t$ trials:

$$P(X \le k_t) \;=\; \sum_{x \le k_t} \frac{\binom{s}{x}\binom{n_b+n_t-s}{n_t-x}}{\binom{n_b+n_t}{n_t}}.$$

The hypergeometric tail is the conditional law of two binomial counts given their combined total: if $K_b \sim \text{Bin}(n_b, p)$ and $K_t \sim \text{Bin}(n_t, p)$ independently, then given $K_b + K_t = s$ the test's count K_t follows this distribution whatever the common p is (§2.5), so the p-value needs no estimate of p.

A test count k_t FAILs iff its p-value is at most α. The p-value is non-decreasing in k_t, so the FAIL region is $\{0, \dots, c-1\}$ and the **integer cutoff** c is the smallest k_t whose p-value exceeds α ($c = 0$ when $k_t = 0$ already exceeds it). The decision is:

$$\text{PASS iff } K_t \ge c.$$

Example ($K_b = 951$, $n_b = 1000$, $n_t = 100$, $\alpha = 0.05$). The p-value of $k_t = 90$ is $0.0347 \le 0.05$ and that of $k_t = 91$ is $0.0720 > 0.05$, so

$$c = 91, \qquad \text{decision: PASS iff } K_t \ge 91,$$

with displayed rate $c/n_t = 0.91$. Two artefacts co-exist and have distinct roles in the report: the integer cutoff $c = 91$, the binding decision rule; and the displayed rate $c/n_t = 0.910000$, a human-readable summary of it.

Structural properties. The cutoff is non-decreasing in K_b: a better baseline never demands less of the test. There is no special case at $K_b = n_b$ — the perfect-baseline map of 1.4.1 is withdrawn (§4.3) — and $c = 0$ at $K_b = 0$ (§4.3.4). The PASS set is always the interval $\{c, \dots, n_t\}$.

Calibration semantics. The random experiment the rule calibrates draws the baseline count and the test count afresh, independently, under one unchanged success probability p: $K_b \sim \text{Bin}(n_b, p)$ and $K_t \sim \text{Bin}(n_t, p)$. The probability of a false degradation signal in that experiment is

$$A(p; n_b, n_t, \alpha) \;=\; \sum_{k_b=0}^{n_b} P_p(K_b = k_b)\, P_p(K_t < c(k_b)),$$

and because the Fisher test is exact conditionally on the pooled total, $A(p) \le \alpha$ at every p. The guarantee is for independent baseline and test samples; running the test over the same list of inputs as the baseline is a design choice outside it, to which the calibration statement does not transfer. The theorem, not a computation, supplies the guarantee at every p. The mavai-R certification is evidence that the reference

implementation realises it: it evaluates $A(p)$ exactly at the points of a fine grid of p from 0.50 to 0.99999, over the operating envelope — 1,922 configurations with n_b from 10 to 10,000, $n_t \leq n_b$ and $n_t/n_b \geq 0.01$, at α = 0.001, 0.01, 0.05 and 0.10 — with a largest value on the grid of 0.965α, 0.976α, 0.982α and 0.986α respectively, and beyond the certified range, at baselines of 20,000 to 100,000 with tests of 1 to 100, with a largest value of 0.97α. As n_b grows at fixed n_t the rule tends to the exact one-sample binomial test at the true rate, which is also at or below α, so no baseline bound and no alpha restriction are needed. For the example above, the size at the assumed common rate — A evaluated as if the unknown common p were the baseline's observed rate — is $A(0.951) = 3.40\%$; it describes the procedure under that assumption, not the run.

A probability computed with the baseline held fixed at its observed rate — here $P_{0.951}(K_t < 91) = 0.024986$ — is a conditional plug-in tail. It may appear in a report only as a labelled diagnostic, never as the procedure's false-alarm probability: it does not describe the experiment the procedure is calibrated for.

Conservatism. The rule is conservative: its size is often below α (median 0.60α to 0.76α at the grid's worst p for each configuration, across the certified grid), and it gives up some power to rules that run closer to α. The certification discloses the cost against the beta-binomial predictive rule of §4.5: across 36,200 compared cells, the predictive rule's power to detect a drop of 0.05 is higher in 33,488, with a median gap of 0.003; it is more than 0.05 higher in 4,012 cells, mostly at $p \geq 0.98$, and the gap is largest at near-perfect small baselines (0.39 at $n_b = n_t = 70$, $\alpha = 0.05$, $p = 0.999$). The predictive rule is not calibrated (it exceeds α in 46 of those cells, up to 1.13α), which is why the methodology does not adopt it.

Design rule. A test planned larger than the baseline it consumes is refused before any sample runs (`TEST_LARGER_THAN_BASELINE`, §5.7.1). The comparison is made once, on the two samplings: the test's planned sample size N_t against the baseline run's sample size N_b, for every criterion the test carries, pass-rate and latency alike. For a pass-rate criterion these are n_t and n_b. The refusal is mavai's design policy, not a statistical necessity: the calculation would remain calibrated, and a test larger than its baseline can gain real power — at $n_b = 100$, a baseline rate of 0.95 and a true test rate of 0.85, $\alpha = 0.05$, the power is 0.708 at $n_t = 100$ and 0.826 at $n_t = 200$. The policy is that the measure experiment producing a baseline is at least as large as any test that consumes it, so a need for a larger test is met by a larger baseline. Larger measure experiments produce more stable references and usually support more sensitive regression tests; the rule nevertheless accounts for the baseline's finite-sample uncertainty directly, and no sample-size ratio substitutes for the calculation.

Epistemic status. Exact result: a conditional exact test, unconditionally conservative.

Integer Cutoffs Are the First-Class Decision Artefact A conformant report for every inferential verdict carries:

- the decision rule and its version (`regression/fisher` v1 or `compliance/exact-binomial` v1);

- configured α;
- the integer decision artefact — the cutoff c (regression) or $k_{\min}$ (compliance);
- the displayed rate c/n_t (regression);
- observed count K and total n, and for regression the baseline's K_b and n_b;
- the calibration statement in one line, naming its random experiment (§7.2);
- power against the declared degradation margin, or the minimum detectable degradation at 80% power where none is declared (§5.3, §5.6).

This requirement applies uniformly to compliance and regression procedures (§3.2), and the transparent-statistics output in §7.1 / §10.2 carries the same fields.

Numerical conventions for the cutoff. The cutoff is found from the p-values directly, compared with α as computed; no intermediate rounding is applied. The displayed rate c/n_t is reported to a stated fixed precision — typically **six decimal places** — for cross-language reproducibility against the mavai-R fixtures; the unrounded c and n_t are retained on the trial record so a downstream consumer can recompute c/n_t at any precision. Two reports implement the same decision iff their integer cutoffs c agree, which is the binding identity. The latency-side analogue of this convention is set out in §12.8.

Conformance Verification The mavai-R project generates reference cutoffs and verdicts. See `inst/cases/threshold_derivation.json` and `inst/cases/regression_decision.json` in the mavai-R repository; the certification surfaces behind the calibration statement are published with each release as a separate asset (§10.6).

3.5 Reference Table: Regression Cutoffs

For a baseline of $K_b = 951$ successes in $n_b = 1000$ trials:

Test samples n_t	Cutoff c at $\alpha = 0.05$	Cutoff c at $\alpha = 0.01$
50	45	43
100	91	89
200	184	181
500	465	460
1000	933	925

Observation: smaller tests have lower displayed rates c/n_t at the same α, because a small test proves less and can demand less.

The canonical cases below show the rule across baseline sizes and at the boundaries, with the 1.4.1 cutoff beside each for comparison:

Case	K_b	n_b	n_t	α	c	c (1.4.1)
Ordinary	951	1000	100	0.05	91	91
Ordinary, larger baseline	1902	2000	100	0.05	91	91
Equal sizes	951	1000	1000	0.05	933	939

Case	K_b	n_b	n_t	α	c	c (1.4.1)
Large baseline, small test	9510	10000	100	0.05	91	91
Near-perfect baseline	99	100	100	0.05	94	96
Perfect baseline	100	100	100	0.05	96	94
Perfect baseline, large	1000	1000	100	0.05	99	97
Zero baseline	0	100	50	0.05	0	0
Small baseline	27	30	25	0.05	18	19
$\alpha = 0.01$	951	1000	1000	0.01	925	933
Test larger than baseline	95	100	200	0.05	`TEST_ LARGER_ THAN_ BASELINE`	184

The near-perfect and perfect rows show the repair of §4.3: 99 of 100 gives a cutoff of 94 and 100 of 100 gives 96, where 1.4.1 gave 96 and then 94.

3.6 Testing Against a Given Threshold (Compliance)

For compliance testing, the threshold is **given**, not derived:

$$p_{\text{req}} = p_{\text{SLA}}$$

There is no experimental baseline—the threshold comes directly from a contract, SLA, SLO, or organizational policy.

Example: A payment processing SLA states ”99.5% transaction success rate.”

- $p_{\text{req}} = 0.995$ (given by contract)
- No $\hat{p}_{\text{baseline}}$ to estimate
- The test asks: does the evidence show $p > 0.995$?

Why compliance testing differs from regression testing Compliance testing is an **affirmative-assurance** procedure. The required rate p_{req} is normative — supplied by an SLA, SLO, policy, or regulatory requirement — and the hypothesis pair is

$$H_0 : p \le p_{\text{req}}, \qquad H_1 : p > p_{\text{req}}.$$

The methodology decides it with `compliance/exact-binomial`, version 1, the exact one-sided binomial test. The smallest passing count is

$$k_{\min} \;=\; \min\bigl\{k : P_{p_{\text{req}}}(K \ge k) \le \alpha\bigr\}, \qquad K \sim \text{Bin}(n, p_{\text{req}}),$$

and the decision is PASS iff $K \geq k_{\text{min}}$. The comparison is inclusive: a count whose upper tail equals α passes. The one-sided Clopper–Pearson lower bound coincides with this test — PASS iff the bound is at least p_{req} — and may be reported beside the verdict as the lower bound. The controlled error event is **false compliance**, and its probability at $p = p_{\text{req}}$ is $P_{p_{\text{req}}}(K \geq k_{\text{min}}) \leq \alpha$ at every n, by construction; at rates below the requirement it is smaller still. Over the certification grid the largest false-compliance probability is 0.99997α.

Example: $p_{\text{req}} = 0.95$, $n = 150$, $\alpha = 0.05$: $k_{\text{min}} = 148$, and the false-compliance probability is 1.815%. With $p_{\text{req}} = 0.99$ and $n = 793$: $k_{\text{min}} = 790$, false compliance 4.369%.

Feasibility. PASS is possible at all only if the all-success outcome passes, $P_{p_{\text{req}}}(K \geq n) = p_{\text{req}}^n \leq \alpha$, that is, only if $n \geq \lceil \log \alpha / \log p_{\text{req}} \rceil$ (§5.7.1). A 99.9% requirement at $\alpha = 0.05$ needs at least 2,995 trials — at 2,995 the design passes only on zero failures ($k_{\text{min}} = 2995$, false compliance 4.996%), and at 2,994 it cannot pass at all. Feasible means PASS is possible, not that the test is adequately powered (§5.5).

Epistemic status. Exact result.

Three regimes need to be kept distinct, because each behaves differently at the boundary $p = p_{\text{req}}$:

1. **Naive observed-rate thresholding.** Compare $\hat{p}$ directly with p_{req}. At $p = p_{\text{req}}$ this produces approximately symmetric pass/fail behaviour from sampling variance; about half of repeated tests yield $\hat{p} \geq p_{\text{req}}$ and would be declared compliant. **It is not the methodology's compliance rule** and is presented here only for contrast.
2. **Exact affirmative compliance** (the methodology's rule). PASS only when $K \geq k_{\text{min}}$. False compliance is at most α. At $p = p_{\text{req}}$ the rule rarely declares compliance, by design.
3. **Failure to demonstrate compliance.** A system whose true success probability is at or modestly above p_{req} will often FAIL the rule. This is not a false positive — FAIL means compliance was not demonstrated, not that the rate is below the requirement. Operators who require a higher chance of demonstrating compliance for a system at a stated rate above the requirement size the test for it (§5.5).

The three operational approaches in §6 — sample-size-first, confidence-first, and threshold-first — are alternative ways to manage that conservatism within regime 2; they do not alter the controlled error event.

Reference Table: Sample Sizes for SLA Verification To demonstrate $p > p_{\text{req}}$ at $\alpha = 0.05$ with power 0.80 when the true rate is $p_{\text{req}} + \delta$ (or, where $p_{\text{req}} + \delta \geq 1$ and the contract declares no alternative rate, the midway rate $(p_{\text{req}} + 1)/2$, which the report names), exact sizing gives (§5.5):

p_{req}	Alternative (true rate)	Required samples
0.90	0.95 ($\delta = 0.05$)	203
0.95	0.97 ($\delta = 0.02$)	694
0.95	0.96 ($\delta = 0.01$)	2,869
0.99	0.995 ($\delta = 0.005$)	2,185
0.995	0.9975 (midway)	4,374
0.999	0.9995 (midway)	not settled within 20,000

Key insight: Higher requirements and smaller margins require dramatically more samples. This is why the confidence-first approach needs an alternative — `minDetectableEffect` or a declared alternative rate: without one, the question "how many samples to demonstrate 99.9%?" has no unique answer. A finite size exists at many alternatives, a true rate of 1 among them; the alternative decides which.

---

4. The Perfect Baseline Problem ($\hat{p} = 1$)

> **Status in this revision.** The perfect baseline no longer needs a special treatment. The 1.4.1 construction replaced a perfect baseline's rate by its Wilson lower bound before deriving the test threshold; that map made the cutoff fall at the last step, so that a perfect baseline demanded less of the test than a near-perfect one (§4.3). `regression/fisher` (§3.4) uses the baseline's counts directly, handles $K_b = n_b$ like any other count, and gives a cutoff that never decreases as the baseline improves. This section keeps its numbering; §4.3 records the withdrawn map in place.

4.1 Problem Statement

A critical pathology arises when the baseline experiment observes **zero failures**:

$$k = n \implies \hat{p} = 1$$

This commonly occurs when testing highly reliable systems (e.g., well-established third-party APIs) where failures are rare but not impossible.

Example: An experiment with $n = 1000$ trials against a payment gateway API yields $k = 1000$ successes.

Why this matters: With $\hat{p} = 1$, naive threshold derivation would set $p^* = 1$, meaning any single failure causes test failure—regardless of sample size or confidence level. This is statistically unsound.

The mavai solution: The Fisher rule of §3.4 needs no special case here. It compares the test with the baseline's counts, and 1000 successes in 1000 trials is evidence of a very high rate, not proof of a perfect one: at $n_t = 100$ and $\alpha = 0.05$ the cutoff is $c = 99$, so a test with one failure passes and a test with two does not.

4.2 Interpretation of 100% Observed Success

An observed rate of $\hat{p} = 1$ from n trials does **not** mean $p = 1$; any one-sided lower confidence bound on p is below 1. The exact (Clopper–Pearson) bound at level $1 - \alpha$ is $\alpha^{1/n}$, and the Wilson bound $n/(n + z^2)$ is a close but slightly optimistic approximation to it. Either is a statement about the procedure across repeated samples, not a posterior probability statement about p given the observed counts.

The Rule of Three (heuristic approximation): with n trials and zero failures, a useful rule of thumb places an approximate 95%-coverage one-sided lower bound at:

$$p \geq 1 - \frac{3}{n}.$$

The Rule of Three is an approximation to the Clopper–Pearson construction in the zero-failure case ($1 - 0.05^{1/n} \approx 3/n$), not an exact result and not a posterior probability statement.

Baseline Samples	95% Lower Bound (Rule of Three)
100	0.970
300	0.990
1000	0.997
3000	0.999

(Side note: this assumes conditions are stable and runs are independent.)

4.3 The Wilson Lower Bound Solution (withdrawn)

This subsection described the 1.4.1 treatment, which is withdrawn as a decision procedure. It computed the one-sided Wilson lower bound of a perfect baseline, $p_0 = n/(n + z^2)$, used p_0 in place of $\hat{p}$ as the effective baseline, and then took the Wilson lower bound of p_0 at the test size as the threshold. Feeding one lower bound into another applied a discount at $k = n$ only, and the cutoff dropped at the last step: 99 of 100 gave 96 while 100 of 100 gave 94, and 999 of 1000 gave 98 while 1000 of 1000 gave 97. The 1.4.1 rule survives in mavai-R only, as `regression/wilson-reference`, to reproduce historical outputs.

Key implementation requirement (unchanged): baselines store (k, n), not merely $\hat{p}$. The Fisher rule uses both counts, and only the counts distinguish a perfect baseline of 100 trials from one of 10,000.

4.3.1 Wilson Lower Bound Formula The general Wilson one-sided lower bound (§2.3.1) is:

$$p_{\text{lower}} = \frac{\hat{p} + \frac{z^2}{2n} - z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

When $\hat{p} = 1$, this simplifies to:

$$p_{\text{lower}} = \frac{n}{n + z^2}$$

It remains a descriptive bound (§2.3.1). It no longer enters any threshold derivation.

4.3.2 Worked Example **Baseline**: $n_b = 1000$ trials, $K_b = 1000$ successes, so $\hat{p}_{\text{baseline}} = 1.0$. **Test**: $n_t = 100$, $\alpha = 0.05$.

The Fisher cutoff (§3.4) is $c = 99$: the test passes with at least 99 successes out of 100. The 1.4.1 construction gave 97 here, through the effective rate $p_0 \approx 0.9973$ and a threshold of ≈ 0.9686.

This is intentionally less strict than requiring 100 successes out of 100. The experiment's perfect observation establishes strong evidence of high reliability, but the test still has sampling variability, and the Fisher rule prices both samples without treating the perfect baseline as proof of $p = 1$. It is stricter than the 1.4.1 answer because 1000 successes in 1000 trials is strong evidence, and the withdrawn map discounted it twice.

4.3.3 Reference Table: Cutoffs for 100% Baselines Each cutoff below is the Fisher cutoff of §3.4 at $\alpha = 0.05$ for a perfect baseline ($K_b = n_b$), with the withdrawn 1.4.1 cutoff beside it:

Baseline n_b	Test $n_t = 50$: c (1.4.1)	Test $n_t = 100$: c (1.4.1)
100	48 (46)	96 (94)
300	49 (47)	98 (96)
1000	50 (48)	99 (97)
3000	50 (48)	100 (98)

Against a perfect baseline of 3000 trials, a test of 100 must itself be perfect: at that baseline size, a single failure in 100 is already evidence of degradation at $\alpha = 0.05$.

4.3.4 The Zero Baseline The mirror of §4.3.2 is the baseline that observed no successes at all: $k = 0$, so $\hat{p}_{\text{baseline}} = 0$. It is stated here because the two boundaries are not symmetric, and the asymmetry decides what an implementation must do.

Threshold derivation degenerates but remains defined. At $K_b = 0$ and $k_t = 0$ the pooled total is $s = 0$, the Fisher p-value is exactly 1, and the cutoff is $c = 0$ at every n_b, n_t and α: the test demands nothing and every outcome clears it. This is the correct reading of the evidence rather than a failure of the construction — a baseline that succeeded on no attempt can demand nothing of its successor. It is nonetheless a degenerate design, and an implementation that derives a cutoff here should say so rather than present 0 as an ordinary acceptance bar. The value is exact; an implementation whose arithmetic produces a cutoff of 1 here has computed the wrong number, and the verdict changes with it.

Sizing is undefined and must be declined. The risk-driven sizing of §5.4.1 asks how large a test must be to reach its target power at a design alternative rate p_{design} below the baseline's. With a baseline rate of 0 there is no rate below the baseline at which to require power, and therefore no sample size that prices the design. The correct response is to decline the design (`ZERO_BASELINE`) and name the criterion and its counts, not to substitute a nominal rate, clamp the baseline rate to a small positive value, or fall back to a fixed-threshold form. Any such repair would price a design against a baseline the evidence does not support, and would do it silently.

The general requirement of §4.3 — that baselines store (k, n) and not merely $\hat{p}$ — is what makes this case distinguishable at all. A baseline reduced to its point estimate cannot tell 0 successes in 10 trials from 0 in $10{,}000$; both present as $\hat{p} = 0$. They are very different states of evidence, and only the counts carry the difference into the report.

4.4 Extended Example: Highly Reliable API

Scenario: Testing a payment gateway integration.

Baseline experiment: 2000 transactions, 0 failures ($\hat{p} = 1$).

Goal: Configure regression tests at $\alpha = 0.05$.

Calculation (Fisher cutoff, §3.4):

1. For a 100-sample test, $c = 100$: the test passes only with 100 successes out of 100.
2. For a 50-sample test, $c = 50$: the test passes only with 50 successes out of 50.

Result: Against 2000 flawless baseline transactions, a single failure in a test of 50 or 100 is already a degradation signal at $\alpha = 0.05$, and the false-degradation-signal probability of the procedure stays at or below 5%. The 1.4.1 construction admitted 98 of 100 and 48 of 50 here, because it discounted the perfect baseline twice (§4.3). A team that wants to tolerate the occasional failure in a small test against so strong a baseline should lower α only with its eyes open: the evidence says the service has almost never failed.

4.5 Theoretical Note: Beta-Binomial Alternative

For statisticians reviewing this methodology: the **Beta-Binomial posterior predictive** approach is a natural alternative to the Fisher rule. It propagates baseline uncertainty into a predictive distribution for future test counts, naturally yields integer thresholds, and — the certification shows — is more powerful than the Fisher rule in most cells (§3.4). The methodology nevertheless decides empirical regression with the Fisher rule because:

- **Calibration.** The Fisher rule's false-degradation-signal probability is at or below α everywhere it has been certified, at every baseline size and every α. The predictive rule with a Jeffreys prior exceeds α in some cells (up to 1.13α in the certification comparison), and its calibration at very large baselines and very small tests would need separate certification.

- **No prior negotiation.** Selecting (a, b) — or defending Jeffreys' $a = b = 0.5$ — is not a conversation most engineering teams can have in the time a regression test is configured. The Fisher rule requires nothing beyond counts and α.
- **Auditability in regulated settings.** An exact frequentist test has a single, externally verifiable interpretation. A posterior predictive bound adds a prior that an auditor must understand, accept, or challenge.
- **Cross-language stability.** Both rules rest on finite sums, but the Fisher rule's hypergeometric tail is available in every statistics library and reproduces exactly across languages.

The trade-off is deliberate: the Fisher rule gives up a little power for exact calibration (median gap 0.003 to the predictive rule; more than 0.05 in about one cell in nine, mostly at near-perfect rates and small baselines). The predictive distribution, for readers who want to compare, is:

$$K_t \mid k, n \sim \text{BetaBinomial}(n_t, a + k, b + n - k)$$

where (a, b) are prior hyperparameters (Jeffreys: $a = b = 0.5$). See Gelman et al. (2013) and Bayarri & Berger (2004) for treatments. The methodology standardises the Fisher rule and the conformance reference data that flows from it; frameworks carry no alternative regression rule.

---

5. Test Execution and Interpretation

Procedure scope of the sections below. §§5.1–5.4 describe the **regression** procedure: the decision rule against the integer cutoff c of §3.4, the Type-I/Type-II error frame in degradation terms, and exact power and sizing against that rule. The compliance procedure's verdict semantics, decision rule ($K \geq k_{\text{min}}$, §3.6), and error event ("false compliance") are stated in §3.2; §5.5 covers compliance sizing. §5.6 (minimum detectable effect) and §5.7 (VERIFICATION/SMOKE intent, the feasibility gate and the configuration errors) apply to both procedures.

5.1 Decision Rule

Given a baseline of K_b successes in n_b trials and a test with n_t samples, with the integer cutoff c derived by the Fisher rule of §3.4:

1. Execute service contract n_t times
2. Count successes K_t
3. Decision (binding, integer-valued form):
   - If $K_t \geq c$: **PASS** (no degradation signal at this cutoff)
   - If $K_t < c$: **FAIL** (degradation signal at α)

The displayed observed rate $\hat{p}_t = K_t / n_t$ is reported alongside the integer count for human readability, and the displayed cutoff c/n_t beside it, but the binding decision is on K_t vs. c. Reporting both prevents auditors and developers from being misled when a displayed-rate boundary lies between two adjacent integer outcomes.

The verdict wording follows the procedure type (§3.2): a **regression** PASS reads "no degradation signal at the configured cutoff," not "equivalence to baseline established"; a **compliance** PASS reads "evidence supports compliance at the configured level," not "no degradation."

5.2 Type I and Type II Errors

	True state: No degradation	True state: Degradation
Test passes	Correct (True Negative)	Type II Error (False Negative)
Test fails	Type I Error (False Positive)	Correct (True Positive)

- **Type I error rate** (α): The probability of a false alarm — the test fails although the service performs as well as ever (the table's False Positive). For the regression procedure the random experiment is the whole comparison: the baseline count and the test count both drawn afresh under one unchanged success probability. In that experiment the Fisher rule's false-alarm probability $A(p)$ is at most α at every p (§3.4), and in practice somewhat below it. The 1.4.1 rule did not have this property: because it treated the baseline's rate as known, its false-alarm probability in the same experiment rose above α, and rose furthest when the test was as large as or larger than the baseline — at $p = 0.95$ and nominal 5%, about 11% with equal sizes (baseline and test of 1000 each) and about 33% with a baseline of 100 and a test of 1000 — while falling far below α when the test was much smaller (0.17% at $p = 0.99$ for a baseline of 1000 and a test of 25).
- **Type II error rate** (β): The probability of a missed alarm — the test passes although the service has genuinely degraded (the table's False Negative). Its complement $1 - \beta$ is the test's **power** (§5.3). Unlike α, β is not a single number for a test; its value depends on:
  - True effect size (how much degradation occurred)
  - Baseline and test sample sizes
  - The cutoff

5.3 Statistical Power

Power is the probability of correctly detecting degradation when it exists:

$$\text{Power} = 1 - \beta = P(\text{Reject } H_0 \mid H_1 \text{ true})$$

The methodology computes power exactly, against the operative cutoff. With the baseline at success probability p_b and the test at $p_t = p_b - \delta$:

$$\text{Power}(\delta) \;=\; \sum_{k_b=0}^{n_b} P_{p_b}(K_b = k_b)\, P_{p_b-\delta}\big(K_t < c(k_b)\big),$$

with $c(k_b)$ the Fisher cutoff of §3.4. It is a finite sum; no normal approximation is involved. A normal-approximation formula prices a different rule and can misstate the operative rule's power by many points.

Example: Detecting a drop from $p_b = 0.95$ to $p_t = 0.90$ with a baseline of $n_b = 1000$ trials and a test of $n_t = 100$ at $\alpha = 0.05$: the exact power is 0.533. With 100 samples, the test detects a 5-percentage-point degradation about half the time. (1.4.1 printed 68% here, from a normal-approximation formula; its own rule's exact power was 54.9%.)

Epistemic status. Exact result under the working model; the baseline rate p_b it takes is the operator's assumption about the service, usually the baseline's observed rate.

Conformance Verification The mavai-R project generates reference power values. See `inst/cases/power_analysis.json` in the mavai-R repository.

5.4 Sample Size for Desired Power

For the regression procedure, sizing asks how large the test must be for the exact power of §5.3 to reach a target at a design alternative rate. There are two sizing operations, and which applies depends on whether the baseline exists yet (§5.4.1):

- **Design sizing**, before the baseline exists: the baseline and the test are planned together — the baseline at a planned size and an expected rate — and the test is sized by the **design power**, which averages over the baseline count yet to be drawn.
- **Resolved sizing**, against an existing baseline: its observed count fixes the cutoff for every candidate test size, and the test is sized by the **resolved power**.

Both find the answer by exact search, subject to the design rule $n_t \leq n_b$.

Example (design sizing): a baseline planned at 2000 trials with an expected rate of 0.95, $\alpha = 0.05$. The smallest test size from which the exact power to detect each true rate stays at the target:

Design alternative rate p_{design}	Power 80%	Power 90%	Power 95%
0.85 (drop of 10 points)	60	83	106
0.90 (drop of 5 points)	200	283	369
0.92 (drop of 3 points)	566	865	1212

5.4.1 Self-Consistent Power for Baseline-Derived Thresholds

A regression test's cutoff is derived at the test's own size, so sizing must be done against the cutoff the test will actually apply, not against a fixed threshold. The methodology does exactly that, in two operations that answer different questions.

Design sizing — before the baseline exists. The baseline and the test are planned together: the baseline at a planned size n_b and an expected rate p_0, the test at the

size to be chosen. For a **design alternative rate** p_{design} — the true rate at which the test must reach its target power; a declared design input, not a measured estimate — the design power is

$$\text{Power}(n_t) \;=\; \sum_{k_b=0}^{n_b} P_{p_0}(K_b = k_b)\, P_{p_{\text{design}}}\big(K_t < c(k_b;\, n_b, n_t, \alpha)\big),$$

read as: the probability that a service whose true rate is p_{design} fails the test — that degradation at least this severe is detected. It is the §5.3 power with $p_t = p_{\text{design}}$. The design alternative rate is not a tolerance: `regression/fisher` has no margin, and the test still flags any degradation from the baseline, including one to a rate above p_{design}; the rate only states where the target power must be reached.

Domain. The construction is defined for $p_{\text{design}} < p_0$ only. A design with $p_{\text{design}} \geq p_0$ asks the test to detect a "degradation" the baseline already exceeds, and is declined (`ALTERNATIVE_NOT_BELOW_BASELINE`); a zero baseline is declined too (`ZERO_BASELINE`, §4.3.4).

The required sample size. Exact power is a sawtooth in n_t, not a smooth increasing curve, because the cutoff moves in integer steps. The required size is therefore the smallest n_t from which power **stays** at or above the target for every larger test up to the baseline size, not the first size at which it touches the target:

$$n_{\text{req}} = \min\big\{n : \text{Power}(m) \geq 1 - \beta \text{ for every } m \text{ with } n \leq m \leq n_b\big\}.$$

Where no test up to n_b reaches and holds the target, the design is declined (`BASELINE_TOO_SMALL`): the planned baseline cannot support the requested power, and a larger measure experiment should be planned.

Resolved sizing — against an existing baseline. Once the baseline has been measured, its observed count K_b fixes the cutoff $c(K_b;\, n_b, n_t, \alpha)$ for every candidate n_t, and the question the developer is asking is about that baseline, not about the baselines it might have been. The test is sized by the resolved power

$$\text{Power}_{\text{resolved}}(n_t) \;=\; P_{p_{\text{design}}}\big(K_t < c(K_b;\, n_b, n_t, \alpha)\big),$$

with the same stays-at rule, the same domain ($p_{\text{design}} < K_b/n_b$) and the same refusals. Once a baseline exists, `BASELINE_TOO_SMALL` is judged by the resolved power: it is declined only when no test up to n_b reaches and holds the target against that baseline.

The two operations can disagree, and each answers its own question. For the §10.3 configuration — 951 of 1000, $\alpha = 0.05$, design alternative rate 0.925, target 0.80 — design sizing for a planned baseline of 1000 at 0.951 declines (the design power is only 0.753 even at $n_t = 1000$), while resolved sizing against the observed 951 admits the test from $n_t = 966$, where the resolved power reaches and holds 0.80 (first reaching it at 868).

The inversion. For a fixed affordable n_t, the detectable rate is the largest p_{design} (the smallest drop from p_0) at which $\text{Power}(n_t) \geq 1 - \beta$, found by bisection on p_{design} over $(0, p_0)$ to an absolute tolerance of 10^{-10}.

Example (design sizing): a baseline planned at 3000 trials with an expected rate $p_0 = 0.87$; the operator requires the target power at a true rate of $p_{\text{design}} = 0.84$; $\alpha = 0.05$, target power $1 - \beta = 0.80$:

$$n_{\text{req}} = 1243, \qquad \text{Power}(1243) = 0.8002.$$

At 891 samples — the answer 1.4.1 gave for this example, from a normal approximation against the withdrawn Wilson floor — the exact power is 0.708. Inverting instead at an affordable $n_t = 100$: the detectable rate is $p_{\text{design}} \approx 0.7701$. With 100 samples, only a collapse of roughly ten percentage points from the 87% baseline is detected with 80% power; smaller degradations pass more often than that.

Scenario walk-through (design sizing). A baseline run of 2000 trials is planned for a service expected to deliver $p_0 = 0.96$. The operator wants a service that has fallen to $p_{\text{design}} = 0.93$ to fail the test at least four times in five; $\alpha = 0.05$, target power $1 - \beta = 0.80$. How many samples must the test run? Watching the cutoff and the power at candidate sizes shows what is being paid for (the cutoff shown is the one a baseline of exactly 1920 successes would give):

candidate n_t	cutoff c	$\text{Power}(n_t)$ at $p_{\text{design}} = 0.93$
50	45	0.19
150	140	0.43
463	436	**0.80**

At $n_t = 50$ a service that has already degraded to the design rate still passes roughly four runs in five. At $n_t = 150$ the sampling noise still hides the degradation more often than not. $n_t = 463$ is the smallest size from which a service truly at 0.93 fails at least four times out of five — the operator's stated risk appetite, priced in samples. The test is not thereby indifferent above 0.93: at $n_t = 463$ a service truly at 0.94 fails about half the time (0.53), and one at 0.95 about one time in five (0.21), because every degradation from the baseline counts as evidence. The same design against a planned baseline of only 300 trials is declined: no test the design rule admits reaches the target. Once the baseline has run — say at 1920 of 2000 — resolved sizing against it gives 460 (its resolved power first reaches 0.80 at 413), and a test of 463 has resolved power 0.813.

Epistemic status. Exact result under the working model.

Conformance Verification The mavai-R project generates reference values for this construction — required sample sizes, power at candidate sizes, detectable-rate inversions and the refusals — as the `risk_driven_sizing` suite in `inst/cases/`.

5.5 Sample Size for SLA Verification

A compliance test demonstrates that the rate exceeds a requirement, so it is sized for the chance of a PASS when the service is in fact better than required. For a requirement p_{req} and a declared assurance margin δ above it, the alternative — the design alternative rate of §5.6 — is the true rate $p_1 = p_{\text{req}} + \delta$; where $p_{\text{req}} + \delta \geq 1$ the margin cannot express an alternative, and the default is the midway rate $p_1 = (p_{\text{req}} + 1)/2$. The contract may instead declare the alternative rate $p_1 \in (p_{\text{req}}, 1]$ directly. The report names the alternative used and its kind — margin, midway or declared — because the power it states is power at that rate. The exact power at size n is

$$P(\text{PASS} \mid p_1) \;=\; P_{p_1}(K \geq k_{\min}(n)),$$

zero wherever the design is infeasible (§5.7.1), and the required sample size is the smallest n from which this power **stays** at or above the target — not the first crossing, since power is a sawtooth in n. It is found by exact search with the feasibility gate inside the search.

Example: a 95% requirement, demonstrated with 80% power when the true rate is 97% ($\delta = 0.02$), at $\alpha = 0.05$: $n = 694$. Power first reaches 0.80 at 601 samples but dips below it again before 694.

Example: a 99.5% requirement at $\alpha = 0.05$ with 80% power. A margin of 1% leaves the unit interval, so the alternative is the midway rate 99.75%, and the exact size is $n = 4374$. The 1.4.1 text sized this requirement at about 477 samples using the opposite (degradation) orientation; at 477 samples a 99.5% requirement cannot PASS at any outcome, since the feasibility minimum is 598.

The reference table in §3.6 gives further sizes.

Key insight: Demonstrating high requirements with small margins requires substantial sample sizes. A 99.9% requirement with its midway alternative does not settle within 20,000 samples at $\alpha = 0.05$ (16,596 at $\alpha = 0.10$): that is expensive, not invalid, and compliance tests have no maximum size.

Epistemic status. Exact result under the working model.

5.6 The Design Alternative Rate

Power-based sizing needs an alternative: the true rate at which the test must reach its target power. The methodology's canonical quantity is that rate, the **design alternative rate** (`designAlternativeRate`, p_{design}). Its meaning depends on the direction of the test:

- For a **regression** test it lies below the baseline rate: the degraded rate the test must detect with the target power (§5.4.1). The test still flags any degradation from the baseline; p_{design} only states where the power is required.
- For a **compliance** test it lies above the requirement: the rate at which a service that does comply must be able to demonstrate it (§5.5). It is an **assurance alternative**, not a degradation.

The `minDetectableEffect` parameter δ is a directional convenience for stating it: for regression the degradation $p_b - p_{\text{design}}$ below the baseline, for compliance the assurance margin $p_1 - p_{\text{req}}$ above the requirement. Where a compliance margin cannot express an alternative ($p_{\text{req}} + \delta \geq 1$) the default is the midway rate, named in the report, and the contract may declare the rate directly (§5.5).

Without an alternative, the question "how many samples to demonstrate $p > 0.999$?" has no unique answer: a finite size exists at many alternatives, and they differ widely.

Mathematical necessity: To detect an arbitrarily small difference from 99.9% would require millions of samples. For infinitesimal effects, infinite samples are required.

Practical reality: No organization needs to detect every possible degradation, or demonstrate compliance for a service only infinitesimally better than required. For regression, there is always a degradation below which a drop does not matter operationally:

System Type	Typical δ	Rationale
E-commerce checkout	1-2%	1% drop = significant revenue loss
Internal tooling	5-10%	User productivity impact
Safety-critical systems	0.1-0.5%	Regulatory requirements
High-frequency trading	0.01%	Financial impact per transaction

When an alternative is required:

In the **Confidence-First approach**, developers must state the alternative — `designAlternativeRate`, or `minDetectableEffect` in its direction — for the framework to compute the required sample size. This applies to both compliance and regression testing.

Without an alternative, no framework can compute a sample size and will use the default sample count instead.

Where no alternative is declared, the report still states what the design can detect. A regression report carries the power at the declared design alternative rate where there is one; where none is declared, it carries the **minimum detectable degradation** at 80% power under the operative rule — the smallest δ at which the §5.3 design power, with the baseline and the test both yet to be drawn, reaches 0.80. For a baseline of 1000 trials at 0.95 and a test of 100 at $\alpha = 0.05$ it is 0.0756: the design detects a drop to about 0.874 with 80% power, and nothing smaller reliably. Once a baseline has been observed, the same inversion can be made on the resolved-test power, with the cutoff fixed; the two need not coincide, and the report says which it inverts (§10.3). No configuration is added; the adequacy statement is always present, so a weak baseline cannot silently produce a meaningless design.

5.7 Test Intent: VERIFICATION vs SMOKE

The mavai methodology distinguishes between two epistemic intentions when running a probabilistic test. The choice of intent governs whether the framework en-

forces the compliance feasibility gate and how verdicts are framed.

Intent	Purpose	Feasibility gate	Verdict language
VERIFICATION	Evidential — produce statistically defensible verdict	Enforced	Full compliance framing
SMOKE	Sentinel — detect gross regressions cheaply	Bypassed	Non-evidential; a normative criterion FAILs only on evidence of non-compliance (§5.7.3)

The default intent is **VERIFICATION**. Developers may opt into SMOKE when appropriate.

5.7.1 The Feasibility Gate (VERIFICATION only) Before any samples execute, the framework checks whether a normative (compliance) test is able to PASS at all. Under the exact rule of §3.6 PASS is possible only if the all-success outcome passes, $p_{\text{req}}^{n} \leq \alpha$:

$$N_{\text{min}} = \left\lceil \frac{\log \alpha}{\log p_{\text{req}}} \right\rceil$$

A feasible design is one in which PASS is possible; it is not thereby adequately powered (§5.5).

Reference table: N_{min} **at** $\alpha = 0.05$

Target (p_{req})	N_{min}	Interpretation
0.50	5	Almost any sample size suffices
0.80	14	Low bar
0.90	29	Moderate
0.95	59	Common threshold — needs at least 59 samples
0.99	299	High reliability — needs substantial samples
0.995	598	At 598 samples PASS requires zero failures
0.999	2,995	Very high reliability
0.9999	29,956	Extreme reliability — impractical for most test suites

(The 1.4.1 gate used the Wilson bound and a rounded z; its minimums — 3, 11, 25,

52, 268, 539, 2,704 and 27,058 — admitted designs, such as 99.5% at 539 to 597 samples, that cannot PASS at any outcome under the exact rule.)

What happens when infeasible: A VERIFICATION test with $N < N_{\text{min}}$ is refused before any sample runs, with the configuration error **`COMPLIANCE_INFEASIBLE`**. The failure message includes:

- The configured sample size and requirement
- The minimum sample size at which PASS is possible
- A suggestion to use SMOKE intent if the test is a sentinel check

This refusal is **distinct from a SUT failure** — it indicates a configuration problem, not a system defect. It is non-ignorable in CI.

Configuration errors. Two configurations are refused before any sample runs, each with a named code in the verdict and the report:

Code	Refused configuration	Intent
`TEST_LARGER_THAN_BASELINE`	A test whose planned sample size N_t exceeds the sample size N_b of the baseline it consumes — judged once on the two samplings, for every criterion the test carries, pass-rate and latency alike; refused by mavai's design policy, not for statistical invalidity (§3.4)	Refused regardless of intent
`COMPLIANCE_INFEASIBLE`	A normative design with $n < N_{\text{min}}$	VERIFICATION; SMOKE runs and is decided by `compliance/smoke-exact-binomial` (§5.7.3)

The latency success counts are never compared with each other for this rule: they are not known before the run, and they cannot exceed N_t. When a test carries both a compliance criterion and a regression criterion and any part is invalid, the whole test is refused; a run never proceeds half-valid. Every applicable code is reported, as a list in the fixed order of the table — `TEST_LARGER_THAN_BASELINE`, then `COMPLIANCE_INFEASIBLE` — so that a configuration with two invalid parts names both and the developer can correct both at once. A valid test with both is decided criterion by criterion and composed as Two Testing Paradigms describes. The existing zero-baseline treatment (§4.3.4) and the latency existence check (§12.5.2.1) are not configuration errors: the latency rank search run before the test, on the expected number of successful latencies, gives a warning and a planning figure, and the one run after it, on the actual count, yields INCONCLUSIVE with `saturated: true` when no rank exists. An explicit latency requirement below its feasibility minimum is refused as `COMPLIANCE_INFEASIBLE` under VERIFICATION when even its planned samples could not pass (§12.3).

Conformance Verification The mavai-R project generates reference feasibility values and refusal cases. See `inst/cases/feasibility.json`, `inst/cases/compliance_decision.json` and `inst/cases/verdict.json` in the mavai-R repository.

5.7.2 SMOKE Intent: When and Why SMOKE tests are appropriate when:

- **Quick feedback** is more valuable than statistical defensibility (e.g. pre-commit hooks, nightly canaries)
- **The sample budget is fixed** by cost constraints and falls below N_{min}
- **The test is exploratory** — developers are still discovering the system's performance characteristics
- **The target is aspirational** — the threshold expresses an SLA that will later be verified with a properly sized test

5.7.3 How SMOKE Decides For **regression** SMOKE tests a statistical asymmetry governs how verdicts should be read:

> A **FAIL** verdict from an undersized regression test is directionally reliable — the test count fell below the cutoff, and the direction of the evidence is clear even if the magnitude is uncertain. A **PASS** verdict from an undersized test provides weak evidence — the test had little power to detect a degradation.

For **normative** (compliance) SMOKE tests the compliance rule of §3.6 cannot serve: below N_{min} no count can PASS, so its FAIL would be certain before the test ran and would carry no evidence about the service. A developer who declares SMOKE has already said that the test is too small to demonstrate the requirement. The question a SMOKE test asks is the sentinel's question: is there evidence that the service falls short? SMOKE therefore reverses the burden of proof, at every sample size, with `compliance/smoke-exact-binomial`, version 1:

$$H_0 : p \geq p_{\text{req}}, \qquad H_1 : p < p_{\text{req}}, \qquad \text{FAIL iff } P_{p_{\text{req}}}(K \leq k) \leq \alpha, \quad K \sim \text{Bin}(n, p_{\text{req}}).$$

The lower tail rises with k, so the failing counts are $0, \ldots, f$, and the binding count is

$$k_{\text{smoke}} \;=\; 1 + \max\{k : P_{p_{\text{req}}}(K \leq k) \leq \alpha\}, \qquad \text{PASS iff } K \geq k_{\text{smoke}},$$

with $k_{\text{smoke}} = 0$ when no count can FAIL. It is the verdict record's `required-pass` under this rule, so early termination reads it like any other rule's: success is guaranteed once K reaches it, and failure is inevitable once the failures exceed $n - k_{\text{smoke}}$. The probability of a FAIL when the service exactly meets its requirement is at most α. Equivalently, FAIL iff the one-sided Clopper–Pearson upper bound $\text{Beta}^{-1}(1-\alpha;\, K+1,\, n-K)$ is at most p_{req}. The exact-boundary convention of §10.6 applies.

Example ($p_{\text{req}} = 0.99$, $\alpha = 0.05$, $n = 100$, a size VERIFICATION refuses, since PASS needs 299): $k_{\text{smoke}} = 97$. 100 of 100 PASSes, and so does 97; 96 FAILs, four misses in 100 being evidence that the rate is below 99%. Two misses in the first 30 of a run

are already evidence of it, $P(K \leq 28) = 0.036$ at $n = 30$; at the five-sample SMOKE default any miss FAILs, $P(K \leq 4) = 0.049$.

A SMOKE PASS is not evidence of compliance, at any size: it states that the run found no evidence of non-compliance. A SMOKE FAIL is evidence of non-compliance at level α. A SMOKE criterion enters neither Type-I envelope of §1.4.6. When $k_{\text{smoke}} = 0$ (rare: a low requirement over very few samples, such as 50% over one) no outcome can FAIL, and the result says so.

Epistemic status. Exact, like §3.6: K is binomial under the i.i.d. Bernoulli model. Reversing the null hypothesis is a different question asked of the same model, not a new method.

The output words each outcome for what it is:

- **SMOKE PASS (regression)**: ”The observed result is consistent with the baseline.”
- **SMOKE FAIL (regression)**: Still clear — ”The observed result is inconsistent with the baseline.”
- **SMOKE PASS (normative)**: ”No evidence that the service falls short of its requirement at this size.”
- **SMOKE FAIL (normative)**: ”Evidence that the service falls short of its requirement.”
- **VERIFICATION PASS (compliance)**: ”Evidence supports compliance with the requirement at the configured level.”
- **VERIFICATION FAIL (compliance)**: ”Compliance with the requirement was not demonstrated at the configured level.”

5.7.4 Intent-Aware Caveats When a SMOKE test runs against a normative threshold (SLA, SLO, or POLICY):

Condition	Caveat
$N < N_{\text{min}}$	”This size cannot demonstrate compliance ($N = x$, need at least y). The SMOKE result reports only whether there is evidence of non-compliance.”
$N \geq N_{\text{min}}$	”PASS is possible at this size. Consider setting `intent = VERIFICATION` for evidential strength, and check the power statement (§5.5).”
$k_{\text{smoke}} = 0$	”FAIL not possible at this size; this result carries no evidence about the service.”

These caveats appear in both summary and verbose output modes.

6. The Three Operational Approaches: Mathematical Formulation

> **Epistemic status**: decision-policy regimes, not distinct inferential theories. The three "approaches" below describe which test-configuration parameters the framework fixes and which it derives, not three different statistical methods. They all sit inside the same Bernoulli/binomial model, and every quantity they derive is computed exactly against the operative decision rules of §3.4 and §3.6.

The canonical names and descriptive glosses are **sample-size-first** (cost-driven), **confidence-first** (risk-driven), and **threshold-first** (externally-dictated); downstream documentation and implementations use these names.

The approach and the threshold origin are independent axes. The approach is which test-configuration parameter the author fixes first. The threshold origin — declared (normative: SLA / SLO / policy) versus derived (empirical, from a measured baseline) — is what selects the compliance or regression procedure and with it the binding decision rule (§1.4.5). Every approach applies to **both** paradigms:

Paradigm	Threshold Source	Symbol Used
Compliance	Given by contract/policy	p_{req}
Regression	Derived from experimental basis	(K_b, n_b)

In practice implementations commonly pair sample-size-first and confidence-first with the empirical origin, and threshold-first with the normative origin. That pairing is conventional, not definitional: §6.3's regression formulation — a declared cutoff judged against a baseline, with the implied α derived — is a legitimate cell of the grid, and is exactly what the `threshold_derivation` fixture's threshold-first cases exercise.

These formulas plan; they never decide. The binding decision rules live elsewhere: §3.4 for the regression procedure (the observed test count against the Fisher cutoff, $K_t \geq c$) and §3.6 for the compliance procedure (the observed count against the exact test's k_{min}). §6 sizes and prices designs against those rules; no verdict is ever computed from it.

Below, we present each approach with formulations for both paradigms.

6.1 Approach 1: Sample-Size-First (Cost-Driven)

Fix the sample count based on budget constraints; compute the implied cutoff or confidence.

Regression Formulation **Given**: $n_t \leq n_b$, α, experimental basis (K_b, n_b)

Compute: the cutoff c by the Fisher rule of §3.4, and the power it affords against a declared degradation, or its minimum detectable degradation (§5.3, §5.6).

Trade-off: Fixed cost; the false-degradation-signal probability is controlled at α; sensitivity is determined.

Compliance Formulation **Given**: n_{test}, p_{req}

With the sample count and the threshold both fixed, the derivable quantities are $k_{\min}$ at the configured α, and whether the design is feasible at all (§5.7.1); after the run, the confidence the evidence achieves is the threshold-first compliance computation of §6.3, which owns it.

6.2 Approach 2: Confidence-First (Risk-Driven)

Fix the confidence and power requirements; compute the required sample size.

Regression Formulation **Given**: α, desired power $(1-\beta)$, design alternative rate p_{design} (or a minimum detectable effect δ, with $p_{\text{design}} = p_0 - \delta$), experimental basis (K_b, n_b) with reference rate p_0

Compute: n_t — the smallest test size from which exact power stays at the target, with $n_t \leq n_b$ (§5.4.1): by design power when the baseline is still being planned, by resolved power against the observed K_b once it exists.

Trade-off: Fixed confidence and detection capability; cost (sample size) is determined.

Compliance Formulation **Given**: α, desired power $(1-\beta)$, minimum detectable effect δ, requirement p_{req}

Compute: n_{test} — by the exact sizing of §5.5 (e.g. a 95% requirement demonstrated with 80% power at a true rate of 97%, at $\alpha = 0.05$, requires 694 samples).

Trade-off: Fixed confidence and detection capability; cost (sample size) is determined.

Critical: An alternative is essential — the `minDetectableEffect` (δ) or a declared alternative rate. Without one, the sizing question has no unique answer (§5.6).

6.3 Approach 3: Threshold-First (Externally-Dictated)

Use an externally-dictated threshold directly; compute the implied α.

Regression Formulation **Given**: n_t, a declared integer cutoff c, experimental basis (K_b, n_b)

Compute: the implied α — the smallest α at which the Fisher rule of §3.4 yields the declared cutoff. The rule yields c exactly when the p-value of $c-1$ is at most α and that of c exceeds it, so the implied α is the p-value of $c-1$ (and 0 for $c = 0$). Where the p-values of $c-1$ and c coincide, no α yields c, and the implied α is undefined.

Example: baseline $K_b = 951$ of $n_b = 1000$, $n_t = 100$. A declared cutoff of 91 has implied $\alpha = 0.0347$. Declaring instead the baseline's own rate as the threshold, 0.951, means a cutoff of 96, whose implied α is 0.555.

Interpretation: A threshold at the baseline's rate fails more than half of all test runs of an unchanged service. A declared cutoff whose implied α exceeds 0.20 is reported as unsound.

Compliance Formulation **Given**: n_{test}, p_{req}

Compute: the implied α of the observed evidence — the exact p-value of the observed count,

$$P_{p_{\text{req}}}(K \geq K_{\text{obs}}),$$

the smallest α at which the exact test of §3.6 would have declared compliance.

Trade-off: Fixed cost and threshold; the confidence the evidence supports is determined after the run — often weak when the true rate is near the requirement.

When to use: Learning the trade-offs, strict compliance requirements where the requirement is non-negotiable.

---

7. Reporting and Interpretation

7.1 Transparent Statistics Output

When transparent statistics mode is enabled, the framework emits a structured report whose canonical, multi-criterion layout — a contract-level composite header followed by one per-criterion analysis block — is specified in §10.2, with a full worked example in §10.3. A single-criterion contract is the $m = 1$ instance of that layout. This subsection defines the per-metric glossary and the p-value alignment rule on which the report's STATISTICAL INFERENCE block depends.

Key Metrics in the Report

Metric	Formula/Value	Interpretation
Decision rule	`regression/fisher` v1, `compliance/exact-binomial` v1, `latency/precedence` v1 or `latency/compliance-exact-binomial` v1	The versioned procedure that produced the verdict
Configured α	α	Error level requested by the developer
Sample size	n (n_t for a regression test)	Number of trials executed
Successes	K (or k)	Number of passing trials
Observed rate	$\hat{p} = K/n$	Point estimate from test
Baseline counts	K_b, n_b	The baseline evidence the regression cutoff uses

Metric	Formula/Value	Interpretation
Integer decision artefact	c (§3.4) or $k_{\min}$ (§3.6)	Binding decision artefact (§5.1)
Displayed cutoff	c/n_t	Cutoff expressed as a rate
Calibration statement	One line naming the random experiment (§7.2)	What the error level controls
Size at the assumed common rate	$A(\hat{p}_b)$ (§3.4)	Informational, regression: the procedure's false-degradation-signal probability were the unknown common rate the baseline's observed rate (≤ α); not a property of the run
False-compliance probability	$P_{p_{\text{req}}}(K \geq k_{\min})$ (§3.6)	Informational, compliance: the error probability of the discrete decision at the requirement (≤ α)
Design power and resolved-test power	§5.3, §5.4.1	Regression, at a design alternative rate: the design power averages over a baseline yet to be drawn; the resolved-test power is the power of this test, whose cutoff the observed baseline has fixed. They answer different questions and are named apart; for a test that consumes an already observed baseline the second is usually the more actionable
Minimum detectable degradation	§5.3, §5.6	The drop detected at the target power; the report says which power it inverts — the design power, or the resolved-test power against the observed baseline — since the two need not coincide
Standard error	$\text{SE} = \sqrt{\hat{p}(1-\hat{p})/n}$	Precision of the estimate
Confidence interval	Wilson score bounds (descriptive), or the Clopper–Pearson lower bound beside a compliance verdict	Range of plausible true values
p-value	tail probability — see paragraph below	Tail probability under the stated null and orientation

P-value alignment. A p-value is meaningful only against a stated null hypothesis and tail; under the methodology, any p-value that appears in a report is the tail probability of the same test and orientation as the decision rule that produced the verdict. The methodology recognises two internally consistent reporting strategies:

1. **The decision rule's own exact p-value** — for regression the one-sided Fisher p-value of the observed test count, $P(X \leq K_t)$ given the pooled total (§3.4), which is at most α exactly when the verdict is FAIL; for compliance the exact binomial p-value $P_{p_{\text{req}}}(K \geq K_{\text{obs}})$, which is at most α exactly when the verdict is PASS.
2. **No p-value**, in which case the report carries the integer decision artefact, the calibration statement, and the raw counts as the decision summary.

Where a p-value is present, the null hypothesis, the alternative, and the tail are part of the same report entry, so that the p-value cannot be read against a different orientation than the one that produced it. A Wald-style z-score is not part of the decision summary; a report that carries one labels it as a diagnostic.

A probability computed with the baseline held fixed at its observed rate — the conditional plug-in tail $P_{\hat{p}_b}(K_t < c)$ — may be shown only as a labelled diagnostic, never as the procedure's false-alarm probability (§3.4).

Example Output A complete worked report is given in §10.3 — the consult-advice tests, which between them exercise both pass-rate decision rules (the regression Fisher cutoff and the compliance exact test). The binding decision summary is always the procedure block together with the observed K and n: a regression verdict is made on K_t versus the integer cutoff c, never on rounded displayed rates.

Conformance Verification The mavai-R project generates reference verdict evaluation values. See `inst/cases/verdict.json` in the mavai-R repository.

7.2 Confidence Statement

A verdict is accompanied by a plain-language confidence statement whose form is determined by the procedure that produced it. Because the two procedures of §3.2 control different error events, the methodology defines two distinct statements, each a frequentist long-run-property statement about the procedure rather than a posterior probability about the individual verdict, and each naming the random experiment it describes.

For the **regression / monitoring** procedure the statement takes the form:

> "This rule was configured at $\alpha = 0.05$. If the baseline and the test were both drawn afresh from a service whose success probability had not changed, and trials are independent and stationary, the procedure would signal degradation with probability at most 5%. This is not a posterior probability that this individual result is wrong."

For the **compliance / assurance** procedure the statement takes the form:

> ”If the true success probability were at or below the requirement, this exact test would falsely declare compliance with probability at most 5%, under the stated model assumptions. This is not a posterior probability that this individual result is correct.”

The distinction matters because the two are not equivalent. The frequentist α level is a property of the procedure under repeated use in a stated random experiment: it bounds the long-run rate at which the procedure produces the controlled error event. It is not the probability, given a particular observed verdict, that that verdict is wrong — a posterior quantity that depends on the prior probability of degradation, the effect-size distribution, and operational context the procedure does not see. Paraphrases such as ”there is a 5% probability that this failure is due to sampling variance rather than actual system degradation” conflate the two and are inconsistent with the frequentist construction used in §3.

7.3 Multiple Testing Considerations

When running multiple probabilistic tests, the per-test α is not the family-wise false-positive probability of the suite. Two distinct families need to be addressed separately, and the methodology takes the dependence-robust bound as its primary statement.

Dependence-robust statement (Bonferroni union bound). For any family of m tests with per-test sizes $\alpha_1, \dots, \alpha_m$, regardless of the dependence structure among the test statistics,

$$P(\text{at least one false positive}) \;\leq\; \sum_{i=1}^{m} \alpha_i.$$

This is the methodology's default family-wise statement. It coincides with the per-criterion Type-I envelope of §1.4.6 specialised to a multi-criterion contract, and it generalises to the cross-suite case below. It is conservative under positive dependence and tight under disjoint rejection regions.

Independent-tests special case. Under mutual independence of the test statistics (rarely true in practice — tests over the same service contract share a baseline, tests in a suite share a serving stack, tests in a CI run are temporally clustered), the bound collapses to

$$P(\text{at least one false positive}) \;=\; 1 - \prod_{i=1}^{m} (1 - \alpha_i).$$

For equal $\alpha_i = \alpha$ this is the familiar $1 - (1-\alpha)^m$. The methodology cites this only to anchor intuition; it does not adopt it as the operational bound, because the independence premise is not generally defensible across the families the framework reports against.

Number of tests	Per-test α = 0.05 (independent)	Per-test α = 0.01 (independent)	Bonferroni envelope α = 0.05	Bonferroni envelope α = 0.01
5	22.6%	4.9%	25.0%	5.0%
10	40.1%	9.6%	50.0%	10.0%
20	64.2%	18.2%	100% (capped)	20.0%

The Bonferroni envelope is the conservative dependence-robust statement; the independent-tests column is the value under an independence model, shown for context. It is not a lower bound: under dependence the probability of at least one false positive can fall below it.

Three families, kept separate. The methodology distinguishes three error-control levels and reports the corresponding α aggregates separately:

- **Per-criterion** α_c — the level configured on a single criterion's inferential test (§1.4.5).
- **Within-contract envelopes**, reported separately by procedure direction (§1.4.6): the **false-compliance envelope** $\sum_{c \in \text{compliance}} \alpha_c$ for compliance criteria and the **false-degradation-signal envelope** $\sum_{c \in \text{regression}} \alpha_c$ for regression criteria. A contract whose criteria are all of one kind reports the corresponding single envelope; a contract that mixes both reports both, labelled by direction.
- **Across-suite envelopes**, likewise split by direction — the family-wise bounds across multiple contracts run in the same suite (a CI build, a release gate): the false-compliance envelope $\sum \alpha_j$ over every compliance decision in the suite (pass-rate and explicit latency requirements), and the false-degradation-signal envelope $\sum \alpha_j$ over every regression decision (pass-rate and baseline-derived latency). The methodology reports them when more than one contract's verdict is bundled into a single suite-level conclusion, and does not add them into one.

Optional FDR-style controls. When the family is exploratory rather than gating — for example, a wide diagnostic sweep across criteria where the operator wants to control the expected proportion of false discoveries rather than the probability of any false discovery — false discovery rate (FDR) procedures may be applied. Two are relevant:

- **Benjamini–Hochberg (BH)**, which controls FDR at the configured level under independence and under the broader class of positive regression dependence on subsets (PRDS). (Benjamini & Hochberg, 1995.)
- **Benjamini–Yekutieli (BY)**, which controls FDR under arbitrary dependence at the cost of a $\sum_{i=1}^{m} 1/i$ penalty. (Benjamini & Yekutieli, 2001.)

An FDR procedure is applied within one family of decisions of one direction: a ”discovery” is either a degradation alert or an affirmative compliance declaration, and the two are not mixed in one procedure. The methodology does not impose FDR control; where it is in use, the choice between BH and BY is part of the report alongside the dependence assumption on which the chosen procedure relies, because the validity of the FDR claim depends on that assumption.

Unadjusted per-test α is a per-test claim. A per-test α reported without any family-wise or FDR adjustment bounds only the per-test Type-I rate; it is not a suite-level claim. Under the methodology such values carry the label **unadjusted per-test** in the report, so that they cannot be read as bounds on the family-wise or false-discovery rate of the suite.

Mitigation options (with their exact target):

Procedure	Controls	Dependence assumption
Per-criterion (no adjustment)	Per-test Type-I rate	None
Bonferroni $\alpha' = \alpha/m$	Family-wise error rate (FWER)	Arbitrary
Benjamini–Hochberg	False discovery rate (FDR)	Independence or PRDS
Benjamini–Yekutieli	False discovery rate (FDR)	Arbitrary
Disclosed $\sum \alpha$ (no rescaling)	FWER as a bound, no per-test rescaling	Arbitrary (this is the §1.4.6 default)

7.4 Threshold Provenance

For auditability and traceability, the methodology records the **source** of the threshold through two attributes:

Attribute	Purpose	Example Values
`thresholdOrigin`	Category of threshold origin	`SLA`, `SLO`, `POLICY`, `EMPIRICAL`
`contractRef`	Human-readable reference to source document	`"SLA v2.1 §4.3"`, `"Policy-2024-Q1"`

Target Source Values

Value	Meaning	Hypothesis Framing
`SLA`	Service Level Agreement (contractual)	”Does the evidence show the SLA requirement is met?”
`SLO`	Service Level Objective (internal target)	”Does the evidence show the SLO target is met?”
`POLICY`	Organizational policy or standard	”Does the evidence show the policy requirement is met?”
`EMPIRICAL`	Derived from baseline experiment	”Has the service degraded from the baseline?”
`UNSPECIFIED`	Not specified (default)	”Does the evidence show the threshold is met?”

Impact on Hypothesis Formulation The `thresholdOrigin` influences how the framework frames the hypothesis test in detailed reports. Normative origins use the affirmative orientation of §3.6 — the null is that the requirement is not met, and a PASS is evidence that it is; the empirical origin uses the degradation orientation of §3.4:

`thresholdOrigin`	H_0 (Null Hypothesis)	H_1 (Alternative)
`SLA`	Rate at or below the SLA requirement	Rate above the SLA requirement (compliance shown)
`SLO`	Rate at or below the SLO target	Rate above the SLO target (compliance shown)
`POLICY`	Rate at or below the policy requirement	Rate above the policy requirement (compliance shown)
`EMPIRICAL`	No degradation from baseline	Degradation from baseline
`UNSPECIFIED`	Rate at or below the threshold	Rate above the threshold

This adaptation ensures that verdicts are framed in the appropriate business context, making reports immediately understandable to stakeholders.

---

8. Assumptions and Validity Conditions

8.1 When Normal Approximation is Valid

The normal approximation to the binomial is adequate when:

$$n \cdot p \geq 5 \quad \text{and} \quad n \cdot (1-p) \geq 5$$

More conservatively (for confidence intervals):

$$n \cdot p \geq 10 \quad \text{and} \quad n \cdot (1-p) \geq 10$$

For p = 0.95:

- Need $n \geq 200$ for conservative criterion

The methodology does not rely on the normal approximation for any decision: descriptive intervals use Wilson at every n (§2.3.1), and decisions, power and sizing are exact (§3.4, §3.6, §5.3–§5.5).

8.2 Independence Violations

If trials are positively dependent — the usual case, as when repeated inputs share a difficulty — the evidence carried by n trials is less than the evidence n independent trials would carry; negative dependence need not reduce it. The reduction is summarised by an effective sample size n_{eff}: the number of independent trials whose evidence is equivalent to the dependent run. When every pair of trials in the run is correlated to the same degree ρ (and $\rho \in [0, 1]$ summarises that shared pairwise correlation), n_{eff} takes the closed form

$$n_{\text{eff}} = \frac{n}{1 + (n-1)\rho}.$$

Detection: Run autocorrelation analysis on trial outcomes in execution order. Significant lag-1 autocorrelation suggests serial dependence — dependence between consecutive trials. It does not diagnose clustering by prompt or by batch, which is detected from the design metadata of §8.2.1.

Mitigation: Increase the number of independent inputs (prompts, or clusters of them) rather than repeating the same ones — more correlated repetitions add false precision, not evidence; reduce repetitions within a cluster; randomise the order of trials, or separate them into batches or time blocks, where the dependence is temporal; otherwise use a design-aware estimator (§8.2.1). Do not sort input samples by difficulty or by any other characteristic: the calculations assume independent samples from one population, and a run that covers only part of a sorted input list is not one — an unchanged service can then appear to degrade.

The n_{eff} expression rests on a strong simplification: it assumes that every pair of trials in the run is correlated to the same degree ρ, regardless of which two trials are chosen. Under that assumption a single number ρ summarises the dependence in the whole run, and the formula converts it into an effective sample size. This is useful as a first-cut sanity check — given a plausible ρ, how badly is the evidence in n trials inflated? — but the assumption rarely holds in probabilistic testing. Real workloads contain pairs of trials that are tightly linked (two invocations of the same prompt, two trials in the same provider batch, two consecutive calls under the same warm cache) alongside pairs that are essentially independent (trials drawn from different prompts, different batches, different time windows). Collapsing that variation into a single ρ understates the structure that matters.

For workloads of this kind — repeated prompts with intrinsically different per-prompt rates, stratified samplings with unequal stratum weights, time-series autocorrelation that decays with lag, batch effects, or nested clustering such as prompt within stratum within batch — the appropriate machinery is the hierarchical model of §8.2.1, generalised estimating equations (Liang & Zeger, 1986), cluster bootstrap, beta-binomial and other hierarchical estimators, or stratified estimators that respect the declared `targetEstimand`. The methodology does not commit to a single correction; it relies on the design metadata of §8.2.1 to make the form of the dependence visible, so that whichever estimator is used carries the assumptions it depends on.

8.2.1 Clustered Sampling Designs Many probabilistic-test workloads, particularly LLM-backed ones, do not present as n i.i.d. invocations of a uniformly-sampled prompt. A common pattern is to run r repetitions of each of J prompts drawn from a curated sampling, giving $n = r \cdot J$ invocations whose evidential content is materially smaller than n independent draws. A natural model is hierarchical:

$$X_{ij} \mid \theta_j \sim \text{Bernoulli}(\theta_j), \qquad \theta_j \sim G,$$

where j indexes prompts (or prompt strata) and G is the prompt-level rate distribution. Under this model, two prompts with intrinsically very different θ_j contribute jointly correlated evidence within each prompt and unequal evidence across prompts; the plain Binomial(n, p) model of §1.1–1.2 mis-states both the variance and the population to which the estimate generalises.

Required design metadata. Every trial record contributing to a baseline or a per-criterion verdict must carry the fields needed to detect and characterise this structure. At minimum:

Field	Purpose
`promptId`	Identifier of the input item under which the trial was generated. Repeated prompts share an ID.
`stratumId`	Stratum identifier when the sampling is stratified (failure mode, language, difficulty, region, …). Optional when the set is unstratified.
`repetitionsPerPrompt`	Number of trials run against each prompt. Reported as a distribution when not constant.
`batchId`	Provider batch / request batch when observable.
`region` / `timeBlock`	Provider region and time bucket when observable; supports detection of batch and temporal effects.
`samplingMode`	One of: with-replacement, without-replacement, exhaustive, adaptive, externally-supplied.
`targetEstimand`	One of: prompt-weighted, call-weighted, production-weighted, severity-weighted, no-generalisation. Fixes which population the per-criterion p_c estimates.
`populationClaim`	One of: `finite-corpus`, `superpopulation`, `no-generalisation` (§8.4.6). Fixes the claim regime of the verdict; a mismatch between baseline and test is a §8.4.5 hard invalidator.

Field	Purpose
`weights`	Prompt, stratum, production, or severity weights, where the target estimand is weighted. Empty when the design is unweighted.
`strata`	Declared strata and their intended population weights, where stratified sampling is used. Empty when the sampling is unstratified.

The estimator must match the declared `targetEstimand`. A call-weighted estimator over a workload with materially uneven repetitions per prompt does not generalise to a prompt-weighted population, and conversely; the methodology does not silently substitute one for the other.

Treatment by intent. When a workload invokes its J distinct inputs repeatedly, so that the invocation count n exceeds J, trials of the same input are not independent draws, and the verdict's calibration under a plain Binomial(n, p) aggregation may no longer be warranted. How much this matters depends on the number of repetitions per input, on how much the inputs differ in difficulty, and on whether the target estimand weights prompts or calls; the methodology sets no numeric trigger. As plain advice: where inputs repeat, record the design in the metadata above and judge whether the clustering is material; where it is, the methodology's response is intent-dependent.

In **SMOKE** mode, the warning of §8.4.4 is sufficient: it qualifies the verdict rather than blocking it, and the operator carries forward an explicitly degraded calibration claim.

In **VERIFICATION** mode, metadata disclosure alone does not restore the i.i.d. premise. The methodology requires one of the following before a population-level VERIFICATION claim is made:

1. an approved clustered, stratified, hierarchical, or design-based estimator matched to the declared `targetEstimand` (for example, a prompt-weighted estimator under the hierarchical model above, GEE per Liang & Zeger 1986, cluster bootstrap, or a beta-binomial estimator) — at which point the calibration claim applies to that estimator, not to the plain binomial decision rule;
2. demotion of the claim, explicitly labelled per §8.4.6 and naming which claim is kept. A realised-run or no-generalisation claim reports that K of these N invocations passed and withdraws the population-level α statement, the interval as an inferential statement and the calibrated reading of the decision rule; a finite-corpus expected-performance claim, about the corpus's expected rate $\bar{\theta}$, still needs repeated or prompt-aware analysis and is not made by relabelling alone;
3. an INCONCLUSIVE verdict or configuration-error treatment for any remaining population-level VERIFICATION claim.

A warning-only treatment of material cluster structure is not by itself sufficient for an affirmative VERIFICATION claim over a named population; under the §8.4.5 severity

policy that cluster structure functions as a hard invalidator unless one of (1)–(3) has been satisfied.

Reference. Liang, K.-Y., & Zeger, S. L. (1986). "Longitudinal data analysis using generalized linear models." Biometrika, 73(1), 13–22.

8.3 Non-Stationarity

Non-stationarity—when the success probability p is not constant—is perhaps the most insidious threat to probabilistic testing. Unlike independence violations, which can sometimes be detected through autocorrelation, non-stationarity may be invisible in aggregate statistics while fundamentally invalidating comparisons.

8.3.1 Forms of Non-Stationarity

Form	Example	Detection Difficulty
Within-experiment drift	Model updates during a long MEASURE run	Moderate (time-series analysis)
Between-experiment drift	System changes between baseline and test	Hard (requires external knowledge)
Contextual variation	Different behaviour on weekdays vs weekends	Easy (if factors are known and tracked)
Gradual degradation	Slow performance decay over months	Hard (no single detectable event)

8.3.2 Statistical Consequences If p changes during the experiment:

- Point estimate $\hat{p}$ reflects time-averaged behaviour, not current behaviour
- Confidence intervals may understate true uncertainty
- Threshold derivations may be based on stale data
- Verdicts may systematically mislead in one direction

If p differs between baseline and test:

- The comparison is between **different populations**
- The hypothesis test answers the wrong question
- Type I and Type II error rates are no longer meaningfully targeted
- Verdicts are statistically meaningless (though they appear valid)

8.3.3 Why This Is Hard Non-stationarity is difficult because:

1. **It's often invisible in aggregate data**: A 95% pass rate could arise from stable 95% behaviour, or from 99% for half the samples and 91% for the other half.
2. **It can occur between experiments**: The system that generated the baseline may literally not exist anymore (different model version, different infrastructure).

3. **It can be caused by external factors**: Changes to dependencies, APIs, or infrastructure that the developer doesn't control or even know about.
4. **The statistical machinery assumes it away**: All the formulas in this document assume p is constant. When it isn't, the formulas still produce numbers—they're just wrong.

8.3.4 The mavai Approach No framework can guarantee stationarity. Instead, the mavai methodology provides **guardrails** that:

1. **Make context explicit**: Covariate declarations force developers to think about what factors might matter.
2. **Make drift visible**: Covariate non-conformance and expiration warnings surface potential violations.
3. **Preserve auditability**: Baseline provenance ensures the conditions of inference are always documented.
4. **Qualify rather than suppress**: Warnings accompany verdicts rather than replacing them.

See Section 8.4 for detailed descriptions of these guardrails.

8.3.5 Developer Responsibilities The framework provides tools; developers must use them wisely:

Responsibility	How the Framework Helps	What Developers Must Do
Identify relevant factors	Standard covariates for common cases	Declare covariates in service contract definitions
Track contextual changes	Automatic covariate resolution and matching	Ensure custom covariates are in environment
Recognise baseline staleness	Expiration warnings	Set appropriate expiration values
Investigate warnings	Clear warning messages with specifics	Don't ignore non-conformance warnings
Refresh stale baselines	Prominent expiration alerts	Run measure experiments when prompted

8.4 Guardrails for Assumption Validity

> **Epistemic status**: diagnostic guardrails / validity aids, not statistical corrections. The mechanisms below (covariates, expiration, provenance, warning semantics) do not repair a violated assumption; they surface its likely presence so an operator can judge whether the verdict is still trustworthy. A test that runs under non-conforming covariates still produces a statistically questionable verdict — the framework just refuses to let that fact be silent.

The statistical validity of probabilistic testing depends on the assumptions outlined in Section 1.3. While no framework can guarantee these assumptions hold, the mavai methodology provides **guardrails**—features that surface violations, qualify results, and encourage practices that preserve statistical validity.

These guardrails embody a key principle: **statistical honesty over silent convenience**. Rather than producing clean verdicts that hide uncertainty, the framework makes the conditions of inference explicit and auditable.

8.4.1 Covariate-Aware Baseline Matching **The problem**: A baseline represents the empirical behaviour of a system under specific conditions. If a probabilistic test runs under different conditions—different time of day, different deployment region, different feature flags—the comparison may be invalid. The samples are drawn from **different populations**.

This is a violation of the **stationarity assumption**: the success probability p is not constant between baseline creation and test execution.

Example: A customer service LLM performs differently during peak hours (high load, queue delays) versus off-peak hours. A baseline measured at 2 AM may not represent behaviour at 2 PM.

The mavai solution: Developers declare **covariates**—exogenous factors that may influence success rates. The exact syntax varies by framework, but the concept is universal across all implementations.

This declaration is an explicit statement: ”These factors may affect performance. Track them.”

How it works:

1. During measure experiments, the framework records the covariate values as part of the baseline specification.
2. During probabilistic tests, the framework resolves the current covariate values and compares them against the baseline.
3. If values differ (non-conformance), the framework issues a **warning** that qualifies the verdict, subject to the severity policy of §8.4.5.

Statistical interpretation: By default, and subject to the severity policy of §8.4.5 — under which an unmatched critical covariate is a major caveat that is an error in VERIFICATION unless explicitly overridden — non-conformance does not change the pass/fail verdict. Instead, it **qualifies** the inference:

> ”Under the assumption that success rates are comparable across these conditions, the test passes. However, this assumption may not hold—the baseline was created on a weekday, but the test is running on a weekend.”

This transforms a hidden assumption into an explicit, auditable caveat.

Why this matters:

Without Covariates	With Covariates
Silent assumption that conditions don't matter	Explicit declaration of relevant conditions
Population mismatch is invisible	Population mismatch triggers warning
False confidence in verdicts	Qualified confidence with documented caveats
Statistical validity unknowable	Statistical validity auditable

8.4.2 Baseline Expiration **The problem**: Systems change over time. Dependencies update, models are retrained, infrastructure drifts. A baseline from six months ago may no longer represent current behaviour—even if all declared covariates match.

This is **temporal non-stationarity**: the success probability p changes over calendar time in ways that cannot be captured by discrete covariates.

The mavai solution: Developers declare a **validity period** for baselines.

This declaration is an explicit statement: ”I believe this baseline remains representative for N days.”

How it works:

1. The expiration value is recorded in the baseline specification along with the experiment end timestamp.
2. During probabilistic tests, the framework computes whether the baseline has expired.
3. As expiration approaches, the framework issues **graduated warnings**:

Time Remaining	Warning Level	Message
> 25% of validity period	None	—
≤ 25%	Informational	”Baseline expires soon”
≤ 10%	Warning	”Baseline expiring imminently”
Expired	Prominent	”BASELINE EXPIRED”

Statistical interpretation: By default, and subject to the severity policy of §8.4.5 — under which a baseline expired beyond a stated grace window is a major caveat that is an error in VERIFICATION unless explicitly overridden — expiration does not change the pass/fail verdict. Instead, it signals:

> ”This baseline is old. The system may have changed in ways not captured by covariate tracking. Interpret results with appropriate caution.”

Complementary to covariates: Covariates catch **known, observable** context changes (weekday vs weekend, region). Expiration catches **unknown, gradual** drift (model updates, dependency changes, infrastructure evolution).

Guardrail	What It Catches	Mechanism
Covariates	Known context mismatch	Explicit factor declaration and matching
Expiration	Unknown temporal drift	Calendar-based validity period

Together, they provide **defence in depth** against non-stationarity.

8.4.3 Baseline Provenance **The problem**: A statistical verdict is only meaningful if its empirical foundation is known. "The test passed" means little without knowing: against what baseline? Under what conditions? With what caveats?

The mavai solution: Every test verdict includes explicit **baseline provenance**:

```
BASELINE REFERENCE
  File:        ShoppingServiceContract-ax43-dsf2.yaml
  Generated:   2026-01-10 14:45 UTC
  Samples:     1000
  Observed rate: 95.1%
  Covariates:  day_of_week=WEEKDAY, time_of_day=08:00/4h, region=EU_CORE
  Expiration:  2026-02-09 (27 days remaining)
```

This ensures that **no inference result is ever detached from its empirical foundation**.

Why this matters for statistical validity:

- Auditors can verify that comparisons are appropriate
- Operators can investigate unexpected results by examining baseline conditions
- Historical analysis can account for which baseline was in effect when
- Reproducibility is supported by explicit documentation

8.4.4 Explicit Warnings Over Silent Failures A consistent design principle across the mavai methodology's guardrails:

> **Warnings qualify verdicts; they do not suppress them** — the default, subject to the severity policy of §8.4.5.

Condition	Default verdict impact (SMOKE, minor caveats)	Under VERIFICATION (§8.4.5)	Warning
Covariate non-conformance	None	Non-critical covariate: none (minor caveat). Critical covariate: error unless overridden (major caveat)	Yes
Expired baseline	None	Near expiration: none (minor caveat). Beyond the grace window: error unless overridden (major caveat)	Yes

Condition	Default verdict impact (SMOKE, minor caveats)	Under VERIFICATION (§8.4.5)	Warning
Multiple suitable baselines	None	None	Yes

This principle reflects a statistical philosophy:

1. **The developer asked a statistical question** (”Does my system meet this threshold?”). The framework answers that question.
2. **The answer may have caveats** (”...but the baseline is old” or ”...but the conditions differ”). The framework surfaces those caveats.
3. **The decision about whether to trust the answer remains with the human**. The framework provides information, not absolution.

This approach preserves statistical honesty without creating operational paralysis. Tests don't mysteriously skip or fail due to metadata issues—they run, and their limitations are documented. The table's default column is not universal: under VERIFICATION intent, §8.4.5 raises hard invalidators to configuration errors and major caveats to errors unless an override is recorded.

8.4.5 Guardrail Severity Levels The ”warnings qualify, do not suppress” principle of §8.4.4 is the right default for SMOKE-intent runs and for diagnostic guardrails whose violation degrades but does not destroy the comparability of the empirical evidence. It is the wrong default for VERIFICATION- intent runs against guardrails whose violation makes the empirical comparison statistically meaningless. A regression FAIL produced by comparing today's behaviour to a baseline of a different model, a different prompt version, a different evaluator, or a different criterion structure is not a regression FAIL at all — it is a configuration error mis-presented as evidence. The methodology classifies guardrails into three severity tiers and binds them to mode-specific behaviour:

Severity	Examples	VERIFICATION behaviour	SMOKE behaviour
Hard invalidator	Structural-reference mismatch (§1.5.2); model-ID mismatch (non-stationarity); material cluster structure (§8.2.1) without an approved estimator matched to the declared `targetEstimand` and without demotion to a no-generalisation or finite-corpus claim (§8.4.6)	Configuration error — no qualified verdict is emitted; the run is rejected.	Warning by default; promotion to error by policy.
Major caveat	Expired baseline beyond a stated grace window; unmatched critical covariate; changed endpoint, region, or serving-stack revision	Default error unless an explicit policy override admits the run with a qualified verdict.	Warning.
Minor caveat	Near-expiration (within graduated-warning band of §8.4.2); non-critical covariate mismatch; near-grace-window batch metadata gaps	Warning.	Warning.

Hard invalidators in VERIFICATION are not qualifying caveats; they are configuration errors. A VERIFICATION run that detects model-ID, structural-reference, or unmitigated material cluster-structure violations terminates with a configuration-error verdict before any statistical inference is attempted. A ”PASS with caveat” or ”FAIL with caveat” verdict against a hard-invalidator violation under VERIFICATION is not produced; the comparison the operator asked the framework to make does not exist under that configuration. SMOKE runs may proceed under operator policy, since SMOKE carries no verification claim (§5.7.2).

Major caveats preserve the option to override the default error under explicit operator policy — useful for short, controlled runs against a baseline that has just expired, for example, where the operator is willing to read the verdict subject to the disclosed caveat. The override must be recorded on the verdict; it is not an implicit relaxation.

Minor caveats retain the §8.4.4 ”qualify, do not suppress” behaviour in both modes. Their statistical impact is bounded; their disclosure is enough.

This stratification prevents the failure mode the §8.4.4 default otherwise admits: a verification report that looks valid, carries an inconspicuous warning footnote, and

reports a regression FAIL whose true cause is that the empirical comparison was no longer between the same two systems.

8.4.6 Population-Claim Discipline: Finite-Corpus vs Superpopulation The binomial, hypergeometric, and order-statistic machinery developed in §§2–4, §7, and §12 produces inferential claims — statements about a population parameter p_c or $Q(p_j)$ that go beyond the sampled trials themselves. An inferential claim is only meaningful relative to a stated population. Every criterion's verdict declares, on the contract, which of three claim regimes its evidence supports. This declaration sits alongside the design metadata of §8.2.1 (`populationClaim`, `samplingMode`, `weights`, `strata`); a mismatch between baseline and test is a §8.4.5 hard invalidator because the two are comparing different estimands.

Finite-corpus claim. The sampling is a fixed, enumerated corpus of inputs, and the criterion's claim applies only to that corpus. Two claims about a corpus must be kept apart, because the service's response to an input is itself random: invoking the same input again can succeed where it failed.

- **Realised-run claim.** Under an exhaustive evaluation, each of the corpus's inputs invoked once, the run's rate K_c/n_c is known exactly — ”the service passed K_c of these n_c invocations” — and there is no uncertainty about that realised observation, so no binomial interval is reported for it.
- **Corpus expected-performance claim.** With θ_j the probability that the service succeeds on input j, the corpus's expected rate is $\bar{\theta} = \frac{1}{J}\sum_{j=1}^{J}\theta_j$. One run is a single realisation of it and does not know it exactly: a claim about $\bar{\theta}$ keeps the response uncertainty, and in general needs repeated invocation of the inputs or a model that accounts for them (§8.2.1).

The hypergeometric distribution describes sampling without replacement only when each item's outcome is fixed before sampling — a corpus of already-labelled items. When each sampled input triggers a new random response from the service, the count of successes is not hypergeometric, and partial evaluation of a corpus against a live service is not modelled by it. Any claim made beyond the corpus is a separate superpopulation claim and must be labelled as such.

Superpopulation claim. The sampling is treated as a sample from a conceptual population — the adversarial-probe distribution, the production input distribution, the red-team-corpus generator, or whichever distribution the criterion intends p_c to characterise. Under a superpopulation claim, the binomial inferential machinery is the right tool, and the decision rules, descriptive intervals, sample-size budgets, and feasibility gates all carry their usual interpretations. The claim is only as strong as the argument that the sample is representative of the named superpopulation; the methodology does not validate this argument, but it does require the named superpopulation to be recorded on the verdict (§1.4.7, `populationClaim` field of §8.2.1).

No-generalisation claim. The trials are convenience evaluations — ad hoc, illustrative, exploratory. No inferential claim is made beyond the specific trials run. PASS / FAIL labels on a no-generalisation criterion describe the run, not a population, and the framework reports them as such. A no-generalisation claim is appropriate for

early-stage development, smoke testing, and diagnostic probes; it is **not** appropriate for compliance, regression, or release-gating evidence.

Where this most often goes wrong: safety-probe and red-team sets. Curated safety-probe and red-team samplings are typically **finite-corpus** by design — the corpus is the artefact, and its inputs are chosen to exercise specific failure modes rather than to represent a production distribution. Reading $\hat{p}_c$ on such a set as a superpopulation rate (production self-harm rate, production jailbreak rate) inflates the claim well beyond what the evidence supports: an adversarially curated probe set is by construction more adverse than the production distribution, and a clean confidence bound over it does not bound the production failure rate. The methodology warns explicitly against this slide: a red-team verdict is a verdict on the red-team corpus, and any extension to production traffic requires a parallel sentinel stream (§1.4.7's clinical-advice example illustrates the pattern). Under the methodology a safety-probe verdict is not labelled superpopulation absent an explicit operator declaration and a recorded representativeness argument; the default classification is finite-corpus, and elevation to superpopulation requires the operator to assume — and the verdict to record — the representativeness claim that elevation depends on.

Operational binding. The three regimes are recorded in the trial record's `populationClaim` field per §8.2.1. A baseline labelled `superpopulation` cannot be used as the reference for a test labelled `finite-corpus` (and vice versa) without the configuration-error treatment of §8.4.5 — the two are not comparing the same object. Mode transitions between baseline and test are likewise hard invalidators in VERIFICATION; an operator who genuinely intends to re-baseline under a new regime starts a new reference-state epoch (§8.5).

8.5 Repeated Use and Sequential Monitoring

The probabilistic-test workflows the methodology targets are not single scientific studies — they are CI builds, nightly runs, release gates, and continuous monitors. A configured per-run nominal α is the calibration of one invocation of the procedure. It is not the long-run false-alarm probability over a horizon of repeated invocations, and it must not be reported as such.

Monitoring horizon. A monitor's calibration is meaningful only relative to a stated horizon H — the number of invocations against which the operator wants the false-alarm probability bounded (tests that share a baseline are not unconditionally independent, §8.5 below). The horizon may be expressed as a number of runs, a wall-clock window (seven days, one quarter), or a number of release gates. A horizon-level claim is meaningful only with respect to a declared H, and under the methodology the horizon is part of the monitor's metadata wherever a horizon-level claim is made.

Per-run nominal α versus horizon-level false-alarm probability. Consider H invocations of a procedure with per-run nominal α against one reference state — for a regression test, one baseline consumed by H tests of an unchanged service. The random experiment the bound speaks of includes the generation of the baseline as well as the H tests, and over it

$$P(\text{at least one false alarm in } H \text{ runs}) \;\leq\; H \cdot \alpha \qquad \text{(Bonferroni)}$$

regardless of the dependence structure across runs, and the expected number of false alarms is $H \cdot A(p) \leq H\alpha$, with $A(p)$ the procedure's unconditional size (§3.4). The bound is **ex ante**: it averages over the baseline that might have been drawn. The Fisher rule gives no uniform α bound conditional on the baseline actually observed. Once a baseline is observed and kept, its cutoff $c(K_b)$ is fixed and each unchanged-service test signals with probability $q(K_b, p) = P_p(K_t < c(K_b))$, which can be far from α in either direction: after a perfect baseline of 100 of 100 (cutoff 96 for a test of 100 at $\alpha = 0.05$), a service still at 0.95 fails 56.4% of the time, although such a baseline had probability 0.0059 and the unconditional size is 2.08%. Because p is unknown, $q(K_b, p)$ cannot be computed; the resolved-test power of §5.3 is the same calculation at a design alternative rate.

The independence expression $1 - (1 - \alpha)^H$ is a generic benchmark only: tests that share one random baseline are not independent even when their own samples are. At $\alpha = 0.05$ and $H = 50$ the Bonferroni bound is 2.5 (i.e., the bound is uninformative, and the expected number of false alarms is at most $H\alpha$ ex ante); the independence benchmark gives $\approx 92\%$. **One historical false alarm in a long-running monitor is therefore not, by itself, evidence of method failure** — it is the expected behaviour of a correctly calibrated test repeated many times. Method failure is diagnosed against an excess rate of false alarms relative to the horizon-level expectation, not against a single occurrence.

Alpha spending and sequential designs (further reading). When horizon-level control matters operationally — a quarterly compliance window, a release-train gate — the sequential-testing literature provides families of procedures that distribute a horizon-level α budget across runs:

- alpha-spending functions (O'Brien–Fleming, Pocock) that allocate the budget across pre-planned interim looks;
- statistical process control / control-chart approaches (Shewhart, CUSUM, EWMA) for continuous monitors;
- group-sequential and always-valid sequential procedures (e.g., mSPRT, confidence sequences).

The methodology does not impose a particular sequential procedure. Operators who require horizon-level guarantees stronger than the Bonferroni $H\alpha$ bound select a procedure whose dependence assumptions match their workflow, and report the horizon-level calibration alongside the per-run calibration.

Baseline refresh as a new reference state. A baseline refresh ends one reference state and begins another, and is recorded as a new reference-state epoch in the verdict stream. The verdict stream is therefore partitioned by epoch, and horizon-level calibrations are computed within an epoch, not across one. Cross-epoch trends are descriptive, not inferential.

Verdict metadata under recurring monitoring. A verdict produced inside a recurring monitor carries, in addition to its per-run nominal α:

- the monitoring horizon H;
- the per-run nominal α;
- the horizon-level false-alarm bound under the stated reference ($H\alpha$ as the dependence-robust statement, ex ante over the baseline's generation and the H tests; the value under an independence model, shown for context, where it is computable);
- the reference-state epoch the verdict belongs to.

A run-level PASS/FAIL stripped of these annotations is not a horizon-level claim; reporting it as one inverts the relationship between per-run and horizon-level calibration.

---

9. Summary of Key Formulas

Every formula below is tagged with its **epistemic status**:

- **Exact** — a theorem under the stated model assumptions.
- **Exact conditional test** — exact given a conditioning statistic, and conservative in the unconditional experiment the procedure is calibrated for.
- **Wilson score construction** — a score-test inversion for the binomial proportion. Not an exact finite-sample procedure; the methodology's default descriptive interval, chosen for its stability near the boundary and good practical coverage relative to Wald. No decision rule uses it.
- **Asymptotic / Normal-approximation** — a planning formula valid when n is large and p is away from 0 and 1. No decision rule or reported power uses it.
- **Heuristic** — a rule of thumb, useful operationally, not a confidence statement.
- **Non-parametric / distribution-free** — exact for any continuous F under i.i.d. sampling.

The decision rules

Rule	Decides	Binding artefact	Epistemic status	Section
`regression/fisher` v1	Empirical regression	Integer cutoff c; PASS iff $K_t \geq c$	Exact conditional test; unconditionally conservative ($A(p) \leq \alpha$)	§3.4
`compliance/exact-binomial` v1	Normative compliance	$k_{\min}$; PASS iff $K \geq k_{\min}$	Exact	§3.6
`latency/precedence` v1	Latency regression	Rank k, or `saturated` (INCONCLUSIVE)	Exact under continuity; conservative under ties	§12.4.2

Rule	Decides	Binding artefact	Epistemic status	Section
`latency/compliance-exact-binomial` v1	An explicit latency requirement	$y_{\min}$; PASS iff $Y \geq y_{\min}$, Y the successful latencies at or below τ	Exact	§12.3.4
`compliance/smoke-exact-binomial` v1	Normative compliance under SMOKE	k_{smoke}; PASS iff $K \geq k_{\text{smoke}}$	Exact	§5.7.3
`latency/smoke-compliance-exact-binomial` v1	An explicit latency requirement under SMOKE	y_{smoke}; PASS iff $Y \geq y_{\text{smoke}}$	Exact	§12.3.4

Estimation (MLE / plug-in estimation)

$$\hat{p} = \frac{k}{n}, \quad \text{SE}(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

The standard error is the plug-in estimate of $\sqrt{p(1-p)/n}$; it is a descriptive quantity, not an exact bound.

Regression Cutoff (Exact conditional test — *`regression/fisher`*)

$$c \;=\; \min\{k_t \in \{0, \ldots, n_t\} : P(X \leq k_t) > \alpha\}, \qquad X \sim \text{Hypergeometric}(n_b{+}n_t,\; K_b{+}k_t,\; n_t)$$

PASS iff $K_t \geq c$. False-degradation-signal probability, baseline and test both random under an unchanged p: $A(p) = \sum_{k_b} P_p(K_b = k_b) P_p(K_t < c(k_b)) \leq \alpha$.

Regression Power (Exact)

Design power, baseline and test both yet to be drawn, at the design alternative rate $p_{\text{design}} = p_b - \delta$:

$$\text{Power}_{\text{design}} = \sum_{k_b} P_{p_b}(K_b = k_b)\, P_{p_{\text{design}}}(K_t < c(k_b))$$

Resolved-test power, the baseline observed at K_b and its cutoff fixed:

$$\text{Power}_{\text{resolved}} = P_{p_{\text{design}}}(K_t < c(K_b))$$

Compliance Decision (Exact — *compliance/exact-binomial*)

$$k_{\min} = \min\{k : P_{p_{\text{req}}}(K \geq k) \leq \alpha\}, \qquad \text{PASS iff } K \geq k_{\min}$$

Equivalently, PASS iff the one-sided Clopper–Pearson lower bound $\text{Beta}^{-1}(\alpha;\, K,\, n - K + 1)$ is at least p_{req}.

SMOKE Compliance Decision (Exact — *compliance/smoke-exact-binomial*)

$$k_{\text{smoke}} = 1 + \max\{k : P_{p_{\text{req}}}(K \leq k) \leq \alpha\} \text{ (0 when none)}, \qquad \text{PASS iff } K \geq k_{\text{smoke}}$$

Equivalently, FAIL iff the one-sided Clopper–Pearson upper bound $\text{Beta}^{-1}(1-\alpha;\, K + 1,\, n - K)$ is at most p_{req}.

Compliance Feasibility (Exact)

$$N_{\min} = \left\lceil \frac{\log \alpha}{\log p_{\text{req}}} \right\rceil$$

Compliance Power and Sizing (Exact)

$P(\text{PASS} \mid p_1) = P_{p_1}(K \geq k_{\min}(n)), \qquad n_{\text{req}} = \text{the smallest } n \text{ from which this stays} \geq 1-\beta$

p_1 is the declared alternative rate, else $p_{\text{req}} + \delta$, else the midway rate $(p_{\text{req}} + 1)/2$; the report names which.

Wald Confidence Interval (two-sided) (Asymptotic — pedagogical, not used by the methodology)

$$\hat{p} \pm z_{\alpha/2} \cdot \text{SE}(\hat{p})$$

Wilson Score Interval (Wilson score construction — default descriptive interval)

$$\frac{\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

At $\hat{p} = 1$ the one-sided lower bound is $n/(n + z^2)$; at $\hat{p} = 0$ it is exactly 0.

Rule of Three, for zero failures (Heuristic — quick approximation at 95% confidence)

$$p \geq 1 - \frac{3}{n}$$

Cross-reference: §1.4.5 admits this as an optional contextual annotation on observational verdicts under an explicitly-stated i.i.d. Bernoulli model. The annotation does not change the observational verdict label.

Sample Size for Precision (Asymptotic — planning approximation based on normal asymptotics)

$$n = \frac{z^2_{\alpha/2} \cdot p(1-p)}{e^2}$$

Empirical Percentile (nearest-rank) (Exact — definition)

$$Q(p) = t_{(\lceil p \cdot n_s \rceil)}, \quad t_{(1)} \leq \cdots \leq t_{(n_s)}$$

Latency Threshold (Non-parametric / distribution-free — *`latency/precedence`*. Exact for continuous F_T; conservative under ties.)

$$r = \lceil P\, n_t/100 \rceil, \qquad \text{breach}(k) = \sum_{j=0}^{r-1} \binom{n_t}{j} \frac{B(k+j,\ n_b - k + 1 + n_t - j)}{B(k,\ n_b - k + 1)}$$

The rank is the smallest $k \leq n_b$ with $\text{breach}(k) \leq \alpha$, and the threshold is $t_{(k)}$, an observed integer-millisecond value. If no $k \leq n_b$ achieves α for the actual number of successful latencies, the result is INCONCLUSIVE with `saturated: true` (§12.4.2).

Explicit Latency Requirement (Exact — *`latency/compliance-exact-binomial`*)

$$Y = \#\{i : t_i \leq \tau\}, \qquad y_{\min} = \min\{y : P_p(Y \geq y) \leq \alpha\}, \quad Y \sim \text{Bin}(n_s, p), \qquad \text{PASS iff } Y \geq y_{\min}$$

---

10. Transparent Statistics Mode

10.1 Purpose

Transparent Statistics Mode exposes the complete statistical reasoning behind every test verdict. This feature serves:

- **Auditors**: Documented proof that testing methodology is statistically sound
- **Stakeholders**: Evidence that reliability claims are justified
- **Educators**: Teaching material for understanding probabilistic testing
- **Regulators**: Compliance documentation for AI system validation

10.2 Output Structure

Under criterion decomposition (§1.4), a run's functional verdict is a structured tuple over the per-criterion verdicts of its contract, all over the run's one sampling, and the test's verdict composes it with the verdict of any latency constraints, over the enforced dimensions only (§12.3.2, §12.6). Transparent Statistics Mode exposes the test verdict at the top, followed by a per-criterion analysis block for each criterion declared on the contract and a latency block for each latency constraint. A view across several tests over different samplings — every test of one service, say — is a reporting matter (§1.4.7): it presents each test's report beside its sampling and forms no composite verdict of its own.

Contract-level header.

Section	Content	Statistical Purpose
Test Verdict	V_{test}, PASS / FAIL / INCONCLUSIVE per §12.3.2 — the structural composite of the enforced dimensions among the functional composite V_{rate} (§1.4.6) and the latency verdict V_{latency} — with the triggering functional criteria and latency constraints named, and V_{rate} and V_{latency} shown beside it, each with its mode (enforced or advisory, §12.6); an advisory dimension is labelled as such and never shown as deciding the test	At-a-glance answer to ”did the test pass?”
Per-Criterion Summary	One line per criterion: verdict (PASS/FAIL/INCONCLUSIVE) and mode (inferential/observational)	Surface every per-criterion outcome on the front page
Latency Summary	One line per latency constraint: percentile and threshold, source (explicit or baseline-derived), decision rule, and verdict (PASS / FAIL / INCONCLUSIVE), under the latency dimension's mode (enforced or advisory, §12.6); an advisory constraint is decided by the same rule and reported the same way, and does not enter V_{test}. Absent when the test asserts no latency constraint	Surface every latency outcome on the front page, beside the criteria

Section	Content	Statistical Purpose
Type-I envelopes	Procedure-direction-specific disclosed envelopes over every binding decision the test makes, in its enforced dimensions: $\alpha_{\text{false-degradation-signal}} \leq \sum_{c \in \text{regression}} \alpha_c + \sum_{j \in \text{baseline-derived latency}} \alpha_j$, and $\alpha_{\text{false-compliance}} \leq \sum_{c \in \text{compliance}} \alpha_c + \sum_{j \in \text{explicit latency}} \alpha_j$, over the enforced dimensions only: an advisory dimension's criteria or constraints contribute nothing (§1.4.6, §12.3.1, §12.6). A test reports each envelope only when it makes decisions of that direction.	The test verdict's family-wise bounds, labelled by procedure direction

Per-criterion analysis block — repeated for each criterion declared on the contract. Each latency constraint, enforced or advisory, has a latency analysis block of the same shape: its rule and α_j, the successful-latency count, the decision artefact ($y_{\min}$ and the within-threshold count for an explicit requirement; the rank, the threshold and the test's percentile, or saturation, for a baseline-derived one), and its verdict.

Section	Content (inferential criterion)	Content (observational criterion)
Hypothesis Test	H_0, H_1, test type, α_c	Mode declaration ("observational"); no H_0/H_1
Observed Data	Denominator: $n_c = N$ (the sampling size, §1.4.5a), K_c, $\hat{p}_c = K_c/n_c$; diagnostics $n_{c,\text{evaluable}}$ and $r_{c,\text{obs}} = n_{c,\text{evaluable}}/n_c$ (the transform/no-value share), and the FAIL breakdown by reason.	Denominator: $n_c = N$, K_c, failures observed; diagnostics $n_{c,\text{evaluable}}$, $r_{c,\text{obs}}$, and the FAIL breakdown by reason.
Threshold Reference	Threshold origin and derivation (see below)	(omitted — no threshold)

Section	Content (inferential criterion)	Content (observational criterion)
Statistical Inference	Decision rule and version, SE, descriptive CI, integer artefact (c or k_{min}), displayed cutoff c/n_t, calibration statement, informational size at the assumed common rate (regression) or false-compliance probability (compliance), design power and resolved-test power named apart (regression) or the sizing alternative and its kind (compliance), minimum detectable degradation, p-value (per §7.1 alignment rule)	(omitted — verdict is deterministic on the observation)
Verdict	Three strands: statistical / observed-rate / operational	Zero-failure assertion with explicit ”no population claim” caveat

The denominator n_c is the size N of the sampling (§1.4.5a): every trial counts, a trial with no testable value being a FAIL (reason: transform/no-value), never excluded.

The **Threshold Reference** section, when shown, adapts to the criterion's origin:

Origin	Content displayed
SLA / SLO / POLICY	Threshold origin, contract reference, normative threshold p_c^*, α_c
EMPIRICAL	Baseline identifier, baseline counts $(K_c^{\text{baseline}}, n_c^{\text{baseline}})$, covariate-match status, test sample size $n_{c,\text{test}}$, derived cutoff c_c and $c_c/n_{c,\text{test}}$ (§1.5.4)

The **inferential verdict's three strands** (per §10.3's example, applied per inferential criterion):

- **Statistical verdict** — the hypothesis-test conclusion under the declared procedure (§3.2). For a **compliance** criterion: PASS iff $K_c \geq k_{\text{min}}$ under the exact one-sided binomial test (§3.6). For a **regression** criterion: PASS iff the observed success count satisfies the integer-cutoff rule $K_c \geq c_c$, where c_c is the Fisher cutoff derived from the baseline's counts at α_c (§3.4); the displayed cutoff and the descriptive Wilson interval are context, not the binding decision.
- **Observed-rate status** — whether the point estimate $\hat{p}_c$ sits on the right side of the displayed threshold. For a **compliance** criterion it can disagree with the statistical verdict — $\hat{p}_c > p_{\text{req}}$ with $K_c < k_{\text{min}}$ — especially near the boundary, and the disagreement is the point of disclosing both. For a **regression** criterion

it cannot: $K_c \geq c_c$ exactly when $\hat{p}_c \geq c_c / n_{c,\text{test}}$, so the strand restates the verdict as a rate for the reader.

- **Operational caution** — what an operator should do next: sample-size adequacy, power against plausible regressions, follow-up recommendations.

Observational criteria do not carry the three strands; their verdict is deterministic on the observation and a single assertion line suffices.

Reproducibility metadata. A transparent-statistics report carries a small reproducibility block, recorded once per contract-level report (not per criterion), naming the numerical ingredients of the verdict that vary across language ecosystems. The block exists so that an auditor or downstream conformance consumer can reproduce the report's numbers exactly against the mavai-R fixtures, and so that divergence is detected early when an implementation upgrades a dependency that perturbs a quantile or a sort.

Field	Content	Purpose
`referenceDataVersion`	The mavai-R fixture release tag consumed at runtime, e.g. `v0.11.0`.	Identifies the conformance oracle the report was checked against.
`quantileSource`	The implementation source of the normal quantile $z_{1-\alpha}$ used in the descriptive Wilson interval — e.g. `R::qnorm`, `Apache Commons Math NormalDistribution`, `statrs::distribution::Normal::inverse_cdf`.	Different libraries can disagree in the last few ulp of $z_{1-\alpha}$; this names the source for cross-language conformance reproduction.
`intervalImplementation`	The Wilson formula variant in use: `closed-form score-inversion` or `iterative` with the solver tolerance.	The closed-form Wilson interval is exact-arithmetic up to floating-point; an iterative variant introduces a tolerance that must be disclosed.

Field	Content	Purpose
`sortStable`	Boolean — whether the sort algorithm used on the latency stream is stable.	Pairs with `tiePolicy`; see §12.8. Stability does not change percentile values but does change which trial-record is bound to a given rank when ties are present.
`tiePolicy`	The tie-handling rule actually used — for the mavai default, `largest-tied-position-for-upper-bound-ranks` (§12.8).	Confirms the implementation is consistent with the conservative-bound rule of §12.8.

These fields are emitted whether or not the contract under test exercises the latency dimension; the latency-specific fields (`sortStable`, `tiePolicy`) carry their declared values even on pass-rate-only contracts so that the report's reproducibility shape does not depend on the contract's content. The reproducibility block is part of the audit trail, not a verdict input — it carries no PASS/FAIL semantics of its own. The forward-looking calibration fixtures of §10.6 are checked against the same `referenceDataVersion`.

10.3 Example output: the consult-advice tests

The consult-advice requirements of §1.4.8 are evidenced over three input populations, so they are exercised by three tests, each bound to its own contract and running over one sampling: an EMPIRICAL inferential criterion over production prompts, an SLO inferential criterion over inputs chosen to elicit clinical terminology, and an observational criterion over adversarial probes. This is the one canonical consult-advice example; §1.4.8 and §1.5.6 describe the same runs, and every number below is generated by the mavai-R oracle (`scripts/companion_numbers.R`). The output opens with a reporting view that lists the three verdicts, then gives each test's report: its test verdict and envelope, then one analysis block per criterion. None of the three tests asserts a latency constraint, so each test verdict is its functional composite (§12.3.2). Inferential blocks carry the three-strand verdict; the observational block reports a single deterministic assertion.

```
CONSULT-ADVICE: REPORTING VIEW ACROSS THREE TESTS
  (each test is one run over one sampling; this view is a summary of
   three verdicts, not a verdict of one run)
═══════════════════════════════════════════════════════════════════════

  Test 1  consult-advice@5                  V_prod        PASS
  Test 2  consult-advice-readability@1      V_complexity  FAIL
  Test 3  consult-advice-self-harm-probe@1  V_probe       PASS

═══════════════════════════════════════════════════════════════════════
TEST 1 of 3: consult-advice@5
═══════════════════════════════════════════════════════════════════════

TEST VERDICT
  Test verdict:               PASS
  Latency:                    none asserted (test verdict = functional)
  Sampling:                   V_prod v5, 1000 samples
  Type-I envelope (§1.4.6):
    False-degradation-signal  ∑ α_c  ≤  0.05    (C_well-formed)

───────────────────────────────────────────────────────────────────────
CRITERION 1 of 1: C_well-formed  (inferential, EMPIRICAL)
───────────────────────────────────────────────────────────────────────

HYPOTHESIS TEST
  procedure:          REGRESSION
  decisionRule:       regression/fisher  v1
  H₀ (null):         no degradation from the EMPIRICAL baseline  (p_t ≥ p_b)
  H₁ (alternative):   degradation  (p_t < p_b)
  Decision:           PASS iff K_t ≥ c  (integer cutoff at α_c = 0.05)
  Sampling:           V_prod  v5  (redacted production prompts, policy XYZ)

OBSERVED DATA
  Attempted trials:           1000
  Evaluable trials:           1000
  Observation rate (r_obs):   1.000
  Effective denominator (n_t): 1000
  Successes (K_t):            953
  Observed rate (p̂_t):       0.953

THRESHOLD REFERENCE
  Threshold origin:     EMPIRICAL
  Baseline ID:          consult-advice@2026-04-01
  Baseline counts:      K_b = 951  over  n_b = 1000  (p̂_b = 0.951)
  Covariate match:      OK  (test indexed at same point in covariate space)
  Test sample size:     n_t = 1000  (≤ n_b)
  Integer cutoff:       c = 933   (one-sided Fisher exact test, §3.4)
  Displayed cutoff:     c/n_t = 0.933
```

```
STATISTICAL INFERENCE
  Standard error:     SE = √(p̂_t(1-p̂_t)/n_t) = √(0.953 × 0.047 / 1000) ≈ 0.00669
   95% Wilson CI:         [0.938, 0.964]   (descriptive)
   Fisher p-value:
     null / tail:         H₀: p_t ≥ p_b;  P(X ≤ 953 | 1904 pooled successes)
     value:               ≈ 0.623
  Calibration:          false-degradation-signal probability ≤ 0.05, with the
                       baseline and the test both random under an unchanged p
   Size at the assumed common rate:
                          A(0.951) ≈ 0.0404   (informational, §3.4: the
                          procedure's size were the common rate 0.951)
  Diagnostic only:     P_{p = 0.951}(K_t < 933) ≈ 0.0048  (baseline held at its
                  observed rate; not the procedure's false-alarm probability)

VERDICT
  Statistical verdict:  PASS
                        Observed successes K_t = 953 meet the integer
                        cutoff c = 933. No degradation signal at
                        α = 0.05.

  Observed-rate status: ABOVE THRESHOLD
                        Observed 0.953 > 0.933. The point estimate and the
                        statistical verdict agree.

  Operational caution:  CHECK POWER
                        At the design alternative rate 0.925:
                          design power         ≈ 0.753  (baseline and test
                                                         both redrawn)
                          resolved-test power  ≈ 0.815  (this baseline's
                                                         cutoff, c = 933)
                        Minimum detectable degradation at 80% power:
                          inverting the design power    0.0278 (to ≈ 0.923)
                          inverting the resolved power  0.0255 (to ≈ 0.925)
                        Resolved sizing against this baseline: 80% power
                        at 0.925 holds from n_t = 966 (§5.4.1).
                        Watch for downward drift over the baseline's
                        validity window.
```

```
TEST 2 of 3: consult-advice-readability@1
```

```
TEST VERDICT
  Test verdict:               FAIL  (triggered by C_layperson-readable)
  Latency:                    none asserted (test verdict = functional)
  Sampling:                   V_complexity v2, 800 samples
  Type-I envelope (§1.4.6):
```

```
    False-compliance          ∑ α_c  ≤  0.001   (C_layperson-readable)

═══════════════════════════════════════════════════════════════════════════
CRITERION 1 of 1: C_layperson-readable  (inferential, SLO)
═══════════════════════════════════════════════════════════════════════════

HYPOTHESIS TEST
  procedure:          COMPLIANCE
  decisionRule:       compliance/exact-binomial  v1
  H₀ (null):          p_c ≤ p_req = 0.98   (compliance not shown)
  H₁ (alternative):   p_c > p_req = 0.98   (compliance shown)
 Decision:       PASS iff K_c ≥ k_min  (exact one-sided binomial test, α_c = 0.001)
  Sampling:         V_complexity  v2  (inputs eliciting clinical terminology)

OBSERVED DATA
  Attempted trials:           800
  Evaluable trials:           800
  Observation rate (r_obs):   1.000
  Effective denominator (n_c): 800
  Successes (K_c):            788
  Observed rate (p̂_c):       0.985

THRESHOLD REFERENCE
  Threshold origin:     SLO
  Contract ref:         Consult-Advice SLO v2 §3.4 (layperson readability)
  Normative threshold:  p_req = 0.98
  Error level:          α_c  = 0.001
  Feasibility:          PASS possible from n = 342; n_c = 800 is feasible
  Smallest passing count: k_min = 796

STATISTICAL INFERENCE
 Standard error:     SE = √(p̂_c(1-p̂_c)/n_c) = √(0.985 × 0.015 / 800) ≈ 0.00430
  Clopper–Pearson lower bound at α_c = 0.001:  ≈ 0.9665
  Wilson lower bound at α_c = 0.001:           ≈ 0.9649   (descriptive)
  p-value:
    method:       exact binomial, H₀: p_c ≤ 0.98 vs H₁: p_c > 0.98
    tail:         P_{p = 0.98}(K ≥ 788)
    value:        ≈ 0.190
  Calibration:          false-compliance probability ≤ 0.001 at p_c ≤ 0.98

VERDICT
  Statistical verdict:  FAIL
                        Observed K_c = 788 is below k_min = 796.
                        Compliance with the SLO was not demonstrated
                        at α = 0.001.

  Observed-rate status: ABOVE THRESHOLD
                        Observed 0.985 > 0.98. The point estimate sits
```

```
                        above the SLO, but the evidence does not
                        demonstrate compliance. The statistical verdict
                        and observed-rate status disagree; the disclosed
                        strands surface the tension that a single
                        PASS/FAIL would hide.

  Operational caution:  INCREASE SAMPLE OR REVISIT α
                        At α_c = 0.001 the requirement is demanding;
                        to demonstrate compliance with power 0.80 for
                        a service truly at 0.985 (alternative: margin
                  0.005), the test needs n_c = 11250 (§5.5). Alternatively re-
examine
                        the contract's choice of α against the
                        consequence the SLO defends against (§1.4.6).

═══════════════════════════════════════════════════════════════════════════
TEST 3 of 3: consult-advice-self-harm-probe@1
═══════════════════════════════════════════════════════════════════════════

TEST VERDICT
  Test verdict:               PASS
  Latency:                    none asserted (test verdict = functional)
  Sampling:                   V_probe v3, 200 samples
  Type-I envelopes:          none  (observational criterion only)

───────────────────────────────────────────────────────────────────────────
CRITERION 1 of 1: C_no-self-harm  (observational)
───────────────────────────────────────────────────────────────────────────

HYPOTHESIS TEST
  Mode:        Observational  (no threshold, no α, no inferential claim)
  Question:    Was any failure of C_no-self-harm observed in the run?
  Sampling:    V_probe  v3  (adversarial self-harm probes,
                             independent of guardrail's training data,
                             per mavai-safety-taxonomy v2)

OBSERVED DATA
  Attempted trials:           200
  Evaluable trials:           200
  Observation rate (r_obs):   1.000
  Effective denominator (n_c): 200
  Successes (K_c):            200
  Failures observed:          0

VERDICT
  Result:      PASS  (observational)
  Statement:   No failure of C_no-self-harm was observed across
               200 probe trials drawn from V_probe v3.
```

```
No population claim: This verdict makes no inferential claim about the
                     population rate of self-harm responses. It is evidence
                       about the service, generator and guardrail together,
                       under adversarial input, not about the guardrail's
                       sensitivity on its own (§1.4.5b).
```

Reading the example.

- Each test's verdict is the composite of its own run; the view across the three is a summary for the reader and pools nothing. The readability test FAILs; the two passing tests are reported in full alongside, and the methodology does not collapse the three to a single label without disclosing each test's evidence.
- The Type-I envelopes are disclosed properties of each test's composite, not control targets (§1.4.6), and are labelled by procedure direction: $\alpha \leq 0.05$ against a false degradation signal in the structural test (C_well-formed) and $\alpha \leq 0.001$ against false compliance in the readability test (C_layperson-readable). The observational criterion contributes to neither.
- C_well-formed and C_layperson-readable demonstrate the two **Threshold Reference** shapes: EMPIRICAL (baseline counts, covariate-match disclosure and the Fisher cutoff) versus SLO (contract-referenced, normative, with $k_{\min}$). The blocks share otherwise-identical structure; only the threshold's provenance and the decision rule differ.
- C_well-formed's calibration line names its random experiment; the plug-in tail, which holds the baseline fixed at its observed rate, is labelled as a diagnostic, and the size at the assumed common rate is labelled as the procedure's size under that assumption. The operational caution names two powers apart at the design alternative rate 0.925: the design power, 0.753, averages over a baseline yet to be drawn, and the resolved-test power, 0.815, is the power of this test, whose cutoff the observed baseline fixed at 933. For a test consuming an already observed baseline the second is usually the more actionable. The minimum detectable degradation is likewise stated for both inversions, since they need not coincide, and sizing against this baseline is resolved sizing (§5.4.1): the test of 1000 is adequate, and 966 would be enough. The 1.4.1 version of this example claimed power 0.80 at 0.925 from a rule that did not have it.
- C_layperson-readable's verdict shows the three strands disagreeing — statistical FAIL, observed-rate above threshold. The disclosure is the point of the three-strand format: an overloaded ”FAIL (close to threshold)” label would lose the information that the point estimate sits above the SLO but the evidence at α = 0.001 does not demonstrate compliance. Its sizing caution names the alternative the power refers to.
- C_no-self-harm omits Threshold Reference and Statistical Inference entirely; its verdict is the zero-failure assertion of §1.4.5, with the explicit non-claim about population rates that the methodology requires of observational verdicts.

10.4 Mathematical Notation

The output uses proper mathematical symbols where terminal capabilities allow:

Concept	Unicode	ASCII Fallback
Sample proportion	$\hat{p}$ (p̂)	p-hat
Population proportion	π	pi
Null hypothesis	H_0	H0
Alternative hypothesis	H_1	H1
Less than or equal	$\leq$	<=
Greater than or equal	$\geq$	>=
Square root	$\surd$	sqrt
Approximately	$\approx$	~=
Alpha (significance)	α	alpha

10.5 Validation by Statisticians

The transparent output enables statisticians to verify:

1. **Hypothesis formulation**: Is the one-sided test appropriate?
2. **Cutoff derivation**: Was the decision rule named in the report applied correctly, and does its calibration statement name the right random experiment?
3. **Confidence interval**: Is the descriptive interval used correctly, and kept apart from the decision?
4. **Sample size adequacy**: Are the caveats about power appropriate?
5. **Interpretation**: Does the plain-English summary accurately reflect the statistics?

10.6 Calibration Conformance

The published mavai-R fixtures (`inst/cases/*.json`, fetched by the frameworks' conformance pipelines) carry **formula-value** and **decision-rule** fixtures: per-input (`inputs, expected`) cases whose `expected` is the output of the companion's formulae and decision rules as computed by R — the cutoff c, $k_{\min}$, the latency rank or its saturation, each configuration error, and the verdicts they compose into. A downstream framework whose integer decision artefacts agree with the fixtures exactly, and whose other outputs agree to the stated tolerance, is in arithmetic step with the oracle. Each fixture file names the decision rules its values depend on and the methodology version it implements.

Arithmetic agreement is necessary, but it does not by itself show that the error rates a procedure claims are the ones it achieves. For that, mavai-R publishes, with each release from fixtures 0.11.0, a separate **calibration certification** asset, never inside the case files. It contains:

- **Regression calibration.** For `regression/fisher`, the exact unconditional false-degradation-signal probability $A(p)$ at each point of a fine grid of p from 0.50 to 0.99999, and its largest value on the grid, for 1,922 configurations of baseline

and test size (n_b from 10 to 10,000, $n_t \leq n_b$, $n_t/n_b \geq 0.01$) at α = 0.001, 0.01, 0.05 and 0.10, and the same beyond the certified range at baselines of 20,000 to 100,000; exact power at drops of 0.01 to 0.10; and, as disclosure, the power of the beta-binomial predictive rule beside it. The certification fails if any value on the grid exceeds α. The grid gives exact values at its points; the guarantee at every p comes from the theorem of §3.4, and the grid is evidence that the reference implementation realises it.

- **Latency calibration.** For `latency/precedence`, the rank, the exact breach probability at it and one rank below, and saturation, for baselines of 10 to 2,000 latencies, tests of no more successful latencies than the baseline's, the four supported percentiles and four α values.
- **Compliance calibration.** For `compliance/exact-binomial`, the false-compliance probability at $p = p_{\text{req}}$ for nine requirements and sizes up to 20,000, and the exact sizing table. `latency/compliance-exact-binomial` applies the same test to the within-threshold count, so the same table certifies it.

Calibration is established at three levels, which are kept apart. The theorems of §3.4, §3.6 and §12.4.2 establish the rules' statistical properties. The mavai-R certification shows that the reference implementation realises them, over the published surfaces and boundary cases. A downstream framework may claim conformance to a rule's calibration provided it implements the named algorithm and passes the prescribed decision-rule fixtures, their exact-boundary cases included; agreement on a finite set of fixtures is evidence that it implements the same rule, not a proof that it does so for every input, and its own tests and review of the implementation carry the rest of that argument. A framework that reports calibration must say which certification asset it relies on.

Composite envelopes. The §1.4.6 envelopes are union bounds over per-criterion sizes that are each at most α_c; they hold under arbitrary dependence and need no separate calibration fixture.

Exact boundaries. Every exact rule compares a probability with α by an inclusive rule: a Fisher p-value at most α FAILs the test count, a binomial upper tail at most α admits the count (compliance and latency compliance), a binomial lower tail at most α FAILs the count (the SMOKE rules), a breach probability at most α admits the latency rank. At an exact boundary the probability equals α, and double-precision evaluation can land on either side of it — the Fisher p-value of 2 test successes in 4 against a baseline of 12 in 12 is exactly 1/20 but computes as slightly above 0.05. Implementations therefore follow one convention: compute the probability with the standard library; when $|\text{value} - \alpha| \leq 10^{-9}\,\alpha$, recompute it exactly in rational arithmetic from the declared inputs, reading declared rates and levels as the exact decimals they are written as (0.05 is 1/20, 0.995 is 199/200); apply the inclusive rule to the exact value. Each probability is a finite sum of rationals — hypergeometric terms for Fisher, binomial terms for the tails, and ratios of factorials for the breach probability, which at the top rank with $r = n_t$ is simply $n_t/(n_b + n_t)$. The mavai-R fixtures carry cases at exact boundaries, named as such, so that an implementation that compares doubles alone fails them.

Two conformance statuses. The methodology keeps two conformance statuses apart:

- **Formula-value conformance.** The implementation's decision artefacts, quantiles and descriptive intervals agree with the mavai-R fixtures within the stated tolerance. This demonstrates arithmetic agreement with the oracle on the inputs covered by the fixtures.
- **Statistical calibration conformance.** The implementation additionally relies on, and names, the published calibration certification of the rules it implements.

A conformant transparent-statistics report distinguishes the two statuses in its metadata. A minimal structural example of the corresponding block is:

```
conformanceStatus:
  formulaValueFixtures: passed
  calibrationFixtures: not-published
  calibrationClaimPermitted: false
```

The example shows a report against a fixture release that predates the certification asset. Against a release that carries it, the same block surfaces its status (`passed`, `failed`, `partial`) and the `calibrationClaimPermitted` flag is set accordingly. The field's purpose is to make any overclaim visible at the report level rather than buried in the implementation's documentation.

10.7 Conformance Checklist

A conformant implementation of this companion is one for which the following hold jointly. The list is descriptive — each item points to the section in which the underlying property is defined — and is intended as an audit summary, not as a re-statement of the methodology.

- Compliance and regression verdicts are kept separate, and each verdict carries the procedure type and the versioned decision rule alongside the null, the alternative, and the decision artefact (§3.2, §7.1).
- Decisions are made on integer artefacts — the Fisher cutoff c (regression) or $k_{\min}$ (compliance) — not on rounded displayed rates (§3.4, §3.6).
- Every error-rate statement names the random experiment it calibrates; a probability computed with the baseline held at its observed rate appears only as a labelled diagnostic, and the size at the assumed common rate is labelled as such (§3.4, §7.1, §7.2).
- A regression report states the design power and the resolved-test power at the design alternative rate, named apart, or the minimum detectable degradation at 80% power where no alternative is declared (§5.3, §5.6); a compliance sizing names its alternative and its kind (§5.5).
- Configurations are refused before any sample runs with the named codes `TEST_LARGER_THAN_BASELINE` (judged on the planned test size against the baseline run's size, for every criterion including latency) and `COMPLIANCE_INFEASIBLE` (under VERIFICATION only); a configuration with an invalid part is refused whole, and every applicable code is reported in that fixed order (§5.7.1).
- A requirement and a baseline on the same postconditions are two criteria — one compliance, one regression — each with one rule, one α, one threshold origin and one verdict, composed by the structural composite (Two Testing Paradigms, §1.4.6).

- When a p-value is reported, it carries its method, null, alternative, and tail, and matches the orientation of the decision rule that produced the verdict (§7.1, §10.2).
- A latency threshold for which no rank achieves α at the actual number of successful latencies is reported as INCONCLUSIVE with `saturated: true`, never as a threshold; before the run the same search gives only a warning and a planning figure (§12.4.2, §12.5.3).
- An explicit latency requirement is decided by `latency/compliance-exact-binomial`, enforced or advisory; the comparison of the observed percentile with the threshold is reported only as a raw figure, labelled as such (§12.3.4).
- Every assertion, functional and latency, normative and empirical, is enforced by default; a run-time switch, never a per-test or per-criterion setting and never an environment setting, makes the functional dimension, the latency dimension or both advisory (§12.6).
- An advisory dimension is decided by its rules and its verdicts are reported, labelled advisory; it never fails the test, adds nothing to the Type-I envelopes, and remains subject to the configuration refusals (§12.6).
- The report's top-level verdict is the test verdict V_{test}: the structural composite of the enforced dimensions among the functional composite and the latency verdict, naming the criteria and latency constraints that decided it; an advisory dimension is reported beside it and never enters it (§12.3.2, §10.2).
- For clustered or repeated-prompt designs, the report either applies an approved estimator matched to the declared `targetEstimand`, or demotes the claim to a named realised-run or finite-corpus claim (withdrawing the calibrated reading, §8.2.1), or returns INCONCLUSIVE for population-level VERIFICATION (§8.2.1, §8.4.6).
- A criterion's denominator is **the full sampling** N (§1.4.5a): a trial is a success only on a clean PASS; a transform/no-value failure is a counted FAIL (carrying that reason), not an exclusion. There is no per-criterion denominator policy and no `CONDITIONAL`/`MARGINAL` enum; no input is filtered out, since every sample is seen by every criterion.
- Availability, when wanted as a distinct metric, is its own criterion; it does not gate or alter another criterion's denominator (§1.4.5a, §10.2).
- The conformance metadata distinguishes formula-value-fixture status from calibration status, and does not claim statistical calibration conformance without naming the certification it relies on (§10.6).
- The transparent-statistics output carries the reproducibility metadata of §10.2 and the numerical-conventions metadata of §12.8.

---

11. Statistical Discipline Through Design

Sections 2–10 developed the statistical machinery in isolation. This short section names the **framework-level disciplines** that bind the machinery into a practice, so that developers who are not statisticians can still produce statistically defensible verdicts.

The disciplines are orthogonal design policies, not theorems:

Discipline	What it surfaces	Where it lives
Explicit thresholds and provenance	Origin of every pass/fail decision	§7.4 threshold provenance, transparent-statistics output
Feasibility gating for affirmative claims	Whether PASS is possible before the test runs	§5.7 VERIFICATION vs SMOKE
Covariate tracking	Non-stationarity across baseline vs. test contexts	§8.4.1
Baseline expiration	Staleness of empirical baselines	§8.4.2
Warnings over silence	Imperfect conditions, not pretending otherwise	throughout
Full audit trail	Reproducibility and post-hoc investigation	§7.4, transparent-statistics mode

Two points are worth naming plainly.

First, these disciplines do not repair violated assumptions. They make assumption drift visible. A test that runs under a stale baseline still produces a statistically questionable verdict — the framework just refuses to let that fact be silent.

Second, ”more samples” is not a substitute for these disciplines. Sample size controls precision and power; it does not create a principled threshold, verify stationarity, or produce an audit trail. The machinery of §§1–10 and the disciplines of §11 are complements, not alternatives.

---

12. Latency: Empirical Percentile Analysis

Epistemic status of this section. The percentile estimator (§12.2.2) is an exact definition, and the precedence rank that decides latency regression (§12.4.2) is an **exact distribution-free result** for i.i.d. samples from a continuous latency distribution: with no degradation, the probability that the test's nearest-rank percentile exceeds the threshold is at most α. The distinction the section keeps is between that predictive threshold for a future test's percentile and an upper confidence bound on the baseline's true quantile — a valid theorem, but not a calibrated test threshold (§12.4.3). An explicit requirement is decided by the exact one-sided binomial test on the count of latencies within the threshold (§12.3.4), also an **exact result**. The feasibility gate (§12.5.3) and the enforcement modes (§12.6) are **design policies**, not theorems.

12.1 The Statistical Challenge of Latency

Pass-rate testing models functional outcomes as Bernoulli trials drawn from a binomial distribution (Section 1). Latency presents a fundamentally different statistical challenge. Service latency distributions are typically:

- **Right-skewed**: A long right tail caused by cache misses, garbage collection pauses, network retransmission, or cold starts
- **Multimodal**: Distinct modes corresponding to fast paths (cached) and slow paths (database lookup, remote API call)
- **Heavy-tailed**: Extreme outliers that are orders of magnitude larger than the median

These characteristics violate the assumptions of parametric models such as the normal distribution. A system with a 200ms mean and 50ms standard deviation could have a p99 of 350ms (near-normal) or 2000ms (heavy-tailed) — the summary statistics cannot distinguish the two.

The mavai approach: Rather than fitting a parametric distribution to latency data, the methodology uses **non-parametric empirical percentiles** exclusively. This distribution-free approach makes no assumptions about the shape of the latency distribution — it characterises the distribution directly from the observed order statistics.

12.2 Empirical Percentile Estimation

12.2.1 Population Definition and the Tripartite Contract A naive single-population view of latency is statistically hazardous: mixing fast failures, slow failures, and successful responses into one distribution describes none of them faithfully. The mavai methodology therefore decomposes the service-level contract into three orthogonal sub-contracts, each with its own estimand and its own inferential machinery:

Sub-contract	Estimand	Statistical treatment
Correctness	$P(\text{semantic success})$	Binomial (§§1–5)
Availability	$P(\text{infrastructure success})$	Binomial (same machinery; currently treated jointly with correctness in the pass-rate dimension)
Latency-given-success	$T \mid S = 1$	Non-parametric empirical percentiles (§12.2.2)

The latency dimension characterises **only the third sub-contract**: the distribution of wall-clock execution time conditional on a successful sample — one whose response is correct. The conditioning event is defined normatively, and it is the same for every contract, with nothing to configure. For the i-th sample let

$$S_i \;=\; \prod_{c=1}^{m} X_{i,c} \;=\; \begin{cases} 1 & \text{if the sample passes every functional criterion,} \\ 0 & \text{otherwise,} \end{cases}$$

the conjunction of the contract's functional criteria (the functional outcome X_i of §1.4.3). Then

$$\mathcal{L} = \{t_i : S_i = 1\}, \qquad T \mid S = 1,$$

where t_i is the execution time of the i-th sample's invocation. A sample that fails for any reason contributes no t_i: a delivery failure — a timeout, a transport failure, no result — and a response that fails any functional criterion — wrong, malformed, unusable for the transform — are both excluded. Throughout this chapter a **successful latency** means the latency of a sample with $S_i = 1$. Let $n_s = |\mathcal{L}|$ denote their number and let $t_{(1)} \le \cdots \le t_{(n_s)}$ be their order statistics. Baseline and test use the same definition, so their latency populations are comparable by construction.

What the conditioning measures: the latency of correct responses, and nothing else. Failed samples produce execution times that are not comparable with the time the service takes to respond correctly. A fast failure (an immediate rejection, $t \approx 0$) and a slow failure (a timeout at $t = 30{,}000$ms) reflect error paths; an incorrect response may have been produced by a path the correct ones do not take. Pooling them would produce percentile estimates that describe none of these populations. The price of conditioning on correctness is that latency is not independent of it: the latency population is only as large as the number of passing samples, and a change in correctness changes which responses are timed.

The perverse-incentive hazard: Because the latency contract is conditional on success, a service could in principle ”improve” its observed $Q_{0.95}$ by converting slow successes into fast failures — moving mass from the latency distribution into the failure column. This is the hazard named when the tripartite contract was introduced (Document History, milestone 3). It is not a defect of the conditioning; it is the reason the three sub-contracts are evaluated **jointly with logical conjunction** (§12.3.2). A service cannot trade correctness or availability for latency under the mavai verdict rule, because every sub-contract must pass independently. Reviewers and auditors should satisfy themselves that the correctness and availability thresholds are tight enough that this trade cannot be exploited silently.

What this is not: The latency distribution treated here is not the user-experienced response-time distribution marginalised over all attempts. A user who receives an error sees no latency value. Organisations that need an unconditional response-time SLA should combine the three sub-contracts explicitly, e.g. by asserting availability at a level sufficient to bound the marginal tail.

12.2.2 Nearest-Rank Interpolation The mavai methodology computes percentiles using the **nearest-rank method**. For a percentile $p \in (0, 1]$ with n_s sorted observations:

$$Q(p) = t_{(\lceil p \cdot n_s \rceil)}$$

where $t_{(k)}$ denotes the k-th order statistic of the sorted sample (one-based), $t_{(1)} \leq t_{(2)} \leq \cdots \leq t_{(n_s)}$. The rank $\lceil p \cdot n_s \rceil$ is clamped to $[1, n_s]$.

Worked example: For $n_s = 200$ successful samples:

Percentile	p	Rank $\lceil p \cdot 200 \rceil$	Order statistic
p50	0.50	100	$t_{(100)}$
p90	0.90	180	$t_{(180)}$
p95	0.95	190	$t_{(190)}$
p99	0.99	198	$t_{(198)}$

Why nearest-rank? Linear interpolation methods (e.g., Type 7 in R) produce fractional percentile estimates. For latency thresholds expressed in integer milliseconds and compared against SLA values, the discrete nearest-rank method produces integer-valued estimates that align naturally with how thresholds are specified and reported.

12.2.3 Summary Statistics In addition to percentiles, the framework reports:

- **Mean**: $\bar{t} = \frac{1}{n_s} \sum_{i=1}^{n_s} t_i$
- **Maximum**: $t_{(n_s)}$

The mean and the percentiles (including the maximum, which is $Q(1.0)$) characterise the distribution shape for reporting purposes. Threshold derivation (§12.4) is non-parametric and operates directly on the order statistics — it does not require a sample standard deviation, and the framework deliberately omits s from the baseline payload to prevent misuse as a summary scale of a distribution that is not well-characterised by its second moment.

12.3 Latency Assertions

12.3.1 The Assertion Model A latency assertion specifies a set of percentile constraints:

$$\mathcal{C} = \{(p_j, \tau_j) : j = 1, \ldots, m\}$$

where p_j is a percentile level (one of $\{0.50, 0.90, 0.95, 0.99\}$) and τ_j is the corresponding threshold in milliseconds, explicit or derived from a baseline (§12.3.3).

Every constraint is decided by the rule for its threshold source: an explicit threshold by `latency/compliance-exact-binomial` (§12.3.4), a baseline-derived threshold by `latency/precedence` (§12.4.2). Its verdict V_j is PASS, FAIL or INCONCLUSIVE

— INCONCLUSIVE when too few successful latencies arrived for any count to pass, when the percentile is degenerate, or when no rank achieves α (§12.5.3). The latency dimension's verdict is formed from the constraints in §12.3.2.

Mode. The latency dimension is **enforced** by default, whatever its constraints' threshold sources: V_{latency} enters the test's verdict and can fail the test. A run-time switch can make the dimension **advisory** (§12.6). An advisory constraint is decided by the same rule and its verdict reported, but V_{latency} then stays out of the test's verdict and cannot fail the test. The mode belongs to the dimension, never to one constraint.

Envelope. Each constraint is decided at its own level α_j. As for functional criteria (§1.4.6), the constraints of an enforced latency dimension disclose a union-bound envelope, split by direction: the probability of one or more false latency degradation signals is at most $\sum_j \alpha_j$ over the baseline-derived constraints, and the probability of one or more false latency compliance claims is at most $\sum_j \alpha_j$ over the explicit constraints. Both bounds hold under arbitrary dependence among the percentiles, which, computed from the same latencies, are strongly dependent. The constraints of an advisory latency dimension contribute to neither, since none of their decisions can fail the test.

12.3.2 Combined Verdict Pass-rate and latency are **distinct** quality dimensions (orthogonal as concerns, not necessarily statistically independent — see §12.7). The test's verdict composes them by the structural rule of §1.4.6, applied twice.

The **latency verdict** is the structural composite of the constraints:

$$V_{\text{latency}} = \begin{cases} \text{PASS} & \text{if every constraint passes,} \\ \text{FAIL} & \text{if any constraint fails,} \\ \text{INCONCLUSIVE} & \text{otherwise.} \end{cases}$$

With V_{rate} the functional dimension's verdict — the structural composite of the test's criteria (§1.4.6) — the **test verdict** is the structural composite of the **enforced** dimensions:

$$V_{\text{test}} = \begin{cases} \text{PASS} & \text{if every enforced dimension is PASS,} \\ \text{FAIL} & \text{if any enforced dimension is FAIL,} \\ \text{INCONCLUSIVE} & \text{otherwise.} \end{cases}$$

Both dimensions are enforced by default (§12.6), and then V_{test} is PASS if V_{rate} and V_{latency} both pass, FAIL if either is FAIL, INCONCLUSIVE otherwise. A test with no functional criteria has $V_{\text{test}} = V_{\text{latency}}$; a test with no latency constraint has $V_{\text{test}} = V_{\text{rate}}$. The mixed outcomes follow: functional PASS with latency FAIL is FAIL, functional PASS with latency INCONCLUSIVE is INCONCLUSIVE, functional INCONCLUSIVE with latency PASS is INCONCLUSIVE, and a FAIL in either dimension outweighs an INCONCLUSIVE in the other. The report names what decided a FAIL or an INCONCLUSIVE: the functional criteria and the latency constraints of the enforced dimensions

with that verdict (§10.2). Configuration refusals (§5.7.1) come before any verdict and are unchanged, whatever the dimensions' modes: a refused test has no V_{test}. This reflects the operational reality that a service must be both correct and responsive. A payment API that succeeds 99.5% of the time but takes 30 seconds for 1% of requests fails its SLA just as surely as one that returns incorrect results.

Advisory dimensions. A run-time switch can make the functional dimension, the latency dimension or both advisory (§12.6). An advisory dimension's verdict is formed exactly as above — each criterion and constraint decided by its rule — and reported beside V_{test}, labelled advisory, but it does not enter V_{test}: with the latency dimension advisory, $V_{\text{test}} = V_{\text{rate}}$; with the functional dimension advisory, $V_{\text{test}} = V_{\text{latency}}$. When no dimension is enforced, the composite is over no verdict and V_{test} is PASS: the test cannot fail on its assertions, and the report states that no dimension bound the verdict. It still errors on a configuration refusal or a defect.

12.3.3 Threshold Sources Latency thresholds can originate from two sources:

Source	Symbol	How specified	Statistical question	Rule
Explicit	τ_j (given)	Annotation or configuration	”Does the system meet the declared latency target?”	`latency/compliance-exact-binomial` (§12.3.4)
Baseline-derived	$\hat{\tau}_j$ (estimated)	Automatic from spec	”Has latency degraded from the measured baseline?”	`latency/precedence` (§12.4.2)

This mirrors the compliance vs. regression dichotomy for pass-rate thresholds (Section 3). Explicit thresholds are normative claims, decided like a pass-rate requirement; baseline-derived thresholds are derived by the precedence rank of §12.4.2, calibrated for the test that will run.

12.3.4 Explicit Requirements: `latency/compliance-exact-binomial` An explicit requirement ”the p-th percentile of successful latencies is at most τ” is the statement $F(\tau) \geq p$, with F the distribution function of a successful latency. Comparing the observed percentile with τ does not support that claim at any stated level: $Q(p) \leq \tau$ holds exactly when at least $\lceil p\, n_s \rceil$ of the n_s latencies are at or below τ, and at the boundary $F(\tau) = p$ that happens often. At p95 over 100 latencies it happens with probability $P(\text{Bin}(100, 0.95) \geq 95) = 0.616$.

The methodology decides an explicit requirement with `latency/compliance-exact-binomial`, version 1: the exact one-sided binomial test of §3.6, unchanged, applied to the count of latencies within the threshold. With

$$Y = \#\{i : t_i \le \tau\}, \qquad H_0 : F(\tau) \le p, \qquad H_1 : F(\tau) > p,$$

the smallest passing count is

$$y_{\min} \;=\; \min\{y : P_p(Y \ge y) \le \alpha\}, \qquad Y \sim \mathrm{Bin}(n_s, p),$$

and the decision is PASS iff $Y \ge y_{\min}$. A latency equal to τ counts as within it. The false-compliance probability is at most α, exactly as for a pass rate, and the exact-boundary convention of §10.6 applies. The Clopper–Pearson lower bound on $F(\tau)$ may be reported beside the verdict.

Example (p95 $\le$ 500 ms, $\alpha = 0.05$, $n_s = 100$ successful latencies): $y_{\min} = 99$. If 96 of the 100 latencies are at or below 500 ms, the observed p95 is within the threshold — the raw comparison passes — but compliance is not demonstrated, and the verdict is FAIL.

How many latencies. The number of successful latencies n_s is known only after the run. PASS is possible only if $n_s \ge \lceil \log\alpha / \log p \rceil$ — 59 at p95 and $\alpha = 0.05$, 299 at p99. A VERIFICATION test whose planned samples are below that minimum is refused before it runs (`COMPLIANCE_INFEASIBLE`), since no count could pass even if every sample succeeded. A run whose actual number of successful latencies falls below it is INCONCLUSIVE: too few latencies to decide.

SMOKE use. Under SMOKE intent an explicit requirement is decided by `latency/smoke-compliance-exact-binomial`, version 1: the reversed rule of §5.7.3 applied to the same count, FAIL iff $P_p(Y \le y) \le \alpha$, PASS iff $Y \ge y_{\text{smoke}} = 1 + \max\{y : P_p(Y \le y) \le \alpha\}$ (0 when none). It decides at any number of successful latencies from one upward, and a run with none is INCONCLUSIVE. At p99 over 100 latencies $y_{\text{smoke}} = 97$: four latencies above τ FAIL. The raw comparison $Q(p) \le \tau$ is reporting context only, under either intent, and is labelled as a raw percentile comparison wherever it appears. It supports no compliance claim. An advisory requirement (§12.6) is not a raw comparison: it is decided by `latency/compliance-exact-binomial` like any other, and only its effect on the test differs.

Epistemic status. Exact result: Y is binomial with success probability $F(\tau)$ for i.i.d. successful latencies, whatever their distribution.

12.4 Threshold Derivation from Baselines

12.4.1 The Problem When a measure experiment records the latency distribution, the observed percentiles are point estimates subject to sampling variability. Using the raw baseline percentile as the threshold would cause frequent false positives — the same problem as using $\hat{p}_{\text{baseline}}$ directly as the pass-rate threshold (Section 3.1).

Example: A baseline observes $Q_{0.95} = 480$ms from $n_b = 935$ successful samples. A subsequent test with $n_t = 192$ successful samples observes $Q_{0.95} = 495$ms. Is this a degradation, or normal variance?

Both percentiles vary: the baseline's, because the baseline is a sample, and the test's, because the test is. The threshold must therefore control the probability that a test's percentile exceeds it when nothing has changed, and it must be derived from a construction that is:

1. **Exact** under the minimal assumptions stated in §12.2.1 (i.i.d. successful latencies).
2. **Distribution-free** — free of any parametric shape or density assumption.
3. **Integer-millisecond** by construction, aligning with how SLA thresholds are expressed and compared.

A construction satisfying all three follows from the rank distribution of order statistics, developed below.

12.4.2 Precedence Rank Latency regression is decided by `latency/precedence`, version 1. Its inputs are the n_b baseline latencies (successful samples), the test size n_t, the percentile level $p \in \{0.50, 0.90, 0.95, 0.99\}$ and α.

The test's rank. The test's nearest-rank percentile is its r-th order statistic, $r = \lceil p\, n_t \rceil$, computed in integer arithmetic as $r = \lceil P\, n_t / 100 \rceil$ with $P \in \{50, 90, 95, 99\}$.

The breach probability. Take the k-th smallest baseline latency $t_{(k)}$ as a candidate threshold. With no degradation — baseline and test latencies i.i.d. from one continuous distribution — the test's percentile exceeds $t_{(k)}$ exactly when fewer than r of the n_t test latencies fall at or below it. The probability of that breach is a finite sum:

$$\text{breach}(k) \;=\; \sum_{j=0}^{r-1} \binom{n_t}{j} \frac{B(k+j,\; n_b - k + 1 + n_t - j)}{B(k,\; n_b - k + 1)}.$$

It follows from the Beta distribution of the order statistic's coverage, $F(t_{(k)}) \sim \text{Beta}(k, n_b - k + 1)$ — the same distribution behind the identity $P(t_{(k)} \geq Q(p)) = P(\text{Bin}(n_b, p) < k)$ — averaged over the binomial count of test values below it. It depends only on ranks, not on the latency distribution.

The rank. $\text{breach}(k)$ decreases as k grows. The operative rank is the smallest $k \leq n_b$ with $\text{breach}(k) \leq \alpha$, and the threshold is $\tau = t_{(k)}$, an observed latency in integer milliseconds. The test PASSes iff its nearest-rank percentile is at most τ; a test percentile equal to the threshold is not a breach.

Existence. If no $k \leq n_b$ achieves $\text{breach}(k) \leq \alpha$, the baseline cannot support a calibrated threshold for a test of this size at this percentile. The result is INCONCLUSIVE, reported with `saturated: true` and no threshold; no rank is clamped to n_b to manufacture one. The decision is made after the run, with n_t the actual number of successful latencies. Before the run the same search on the expected number gives a warning and a planning figure (§12.5.2.1, §12.5.3); the expected number is not a lower bound, so the pre-run result decides nothing.

Properties:

- **Integer-ms by construction**: τ is an observed latency. No rounding, no ceiling, no artefacts.
- **Monotone in** α: a smaller α gives a higher rank and hence a looser (more conservative) threshold.
- **Depends on the test size**: a smaller test's percentile is more variable, so it needs a higher rank; the threshold is derived for the test that will run, as the regression cutoff is (§3.4).
- **Calibrated**: with no degradation, the probability that the test breaches the threshold is $\text{breach}(k) \leq \alpha$, exactly; the certification evaluates it at the rank over 1,792 cells of baseline size, test size, percentile and α, as evidence that the reference implementation realises the result (§10.6).

Continuity and ties. The exactness argument above assumes a continuous latency distribution. In practice, wall-clock latencies are reported in integer milliseconds, which induces ties; with ties, a test percentile equal to the threshold is not a breach, and the breach probability can only fall. The threshold stays valid and becomes **conservative**, not anti-conservative. Heavy ties — a mock service recording 0 ms for every call, say — are stated in the report. Practitioners who care about the tightness gap should report latencies at higher resolution (microseconds) before applying the construction.

Epistemic status. Exact result under continuity; conservative under ties.

12.4.3 Statistical Interpretation A test with observed $\hat{Q}_{\text{test}}(p) \leq \tau$ means: the test's percentile is consistent with no degradation from the baseline, at level α.

A breach ($\hat{Q}_{\text{test}}(p) > \tau$) means: the test's percentile exceeds a threshold that, with no degradation, it would exceed with probability at most α — evidence of latency degradation at that level.

Confidence versus prediction. Until milestone 16 the threshold was the binomial order-statistic upper confidence bound on the baseline's true quantile: $\tau = t_{(k)}$ with k the smallest rank at which $P(\text{Bin}(n_b, p) \geq k) \leq \alpha$. That construction is a valid theorem about the baseline's quantile, and it remains available as a descriptive bound, but it is not a calibrated threshold for a future test's percentile: it ignores the test's own sampling variation, and its no-degradation breach probability was uncontrolled — 22.3% in the §12.4.5 example at nominal 5%, and worst when the test is small relative to the baseline. The precedence rank answers the question the test asks. The 1.4.1 construction survives in mavai-R only, as `latency/order-statistic-bound`, to reproduce historical outputs.

12.4.4 Supporting Comparison: Bootstrap (withdrawn) This subsection compared the withdrawn order-statistic confidence bound of §12.4.3 with a percentile bootstrap, as an illustration for engineering readers. The comparison concerns a bound that no longer decides anything, and its fixture (`latency_threshold_bootstrap`) is no longer published; `scripts/bootstrap_compare.R` in mavai-R still reproduces the table for historical reference. A standard-error scaling such as $s/\sqrt{n_s}$ would understate tail-percentile uncertainty on heavy-tailed distributions by a factor that grows

with skewness; the rank-based construction of §12.4.2 has no such defect.

12.4.5 Worked Example **Baseline**: $n_b = 935$ successful samples — the seeded lognormal baseline of the mavai-R `latency_threshold` fixture, whose nearest-rank p95 is $Q_{0.95} = 787$ms. **Test**: $n_t = 192$ successful samples, p95, $\alpha = 0.05$.

The test's nearest-rank percentile is its $r = \lceil 95 \cdot 192/100 \rceil = 183$rd order statistic. The breach probability falls below α first at rank 911:

$$\text{breach}(910) \approx 5.94\% > 5\%, \qquad \text{breach}(911) \approx 4.95\% \leq 5\%.$$

So the threshold is $\tau_{0.95} = t_{(911)} = 875$ms, the 911th-smallest baseline latency — an observed value in milliseconds, by construction. The baseline's own p95 is its $\lceil 0.95 \cdot 935 \rceil = 889$th value, so the threshold sits 22 ranks above the baseline percentile.

A subsequent test with $\hat{Q}_{0.95,\text{test}} \leq t_{(911)}$ passes; a test percentile above $t_{(911)}$ breaches the threshold and constitutes evidence of degradation at $\alpha = 0.05$. The 1.4.1 construction used rank 900 here, whose no-degradation breach probability for a 192-sample test is 22.34%.

12.5 Sample Size Requirements for Percentile Estimation

12.5.1 The Problem An empirical percentile $Q(p)$ is computed from the $\lceil p \cdot n_s \rceil$-th order statistic. When n_s is small relative to p, the estimate is unreliable:

- For $p = 0.99$ with $n_s = 10$: $\lceil 0.99 \times 10 \rceil = 10$ — the ”99th percentile” is simply the maximum value. A single outlier determines the result.
- For $p = 0.99$ with $n_s = 50$: $\lceil 0.99 \times 50 \rceil = 50$ — still the maximum. The p99 only becomes distinct from the maximum when $n_s \geq 100$.

12.5.2 Minimum Sample Sizes The methodology enforces minimum sample counts for each percentile level based on the requirement that the percentile estimate be based on at least one observation below it in the sorted order:

Percentile	p	Minimum n_s	Rationale
p50	0.50	5 (engineering)	$\lceil 0.50 \cdot 5 \rceil = 3$: 3rd order statistic of 5; two values below, two above. The mathematical minimum is 3 (the 2nd of 3, one value on either side); 5 is kept because a median of 3 or 4 values is too fragile to act on
p90	0.90	10	$\lceil 0.90 \cdot 10 \rceil = 9$: 9th order statistic of 10; one value above
p95	0.95	20	$\lceil 0.95 \cdot 20 \rceil = 19$: 19th order statistic of 20; one value above

Percentile	p	Minimum n_s	Rationale
p99	0.99	100	$\lceil 0.99 \cdot 100 \rceil = 99$: 99th order statistic of 100; one value above. Below 100, p99 = max

These thresholds ensure that the percentile estimate is not degenerate (i.e., not simply the minimum or maximum of the sample); for p90, p95 and p99 they are the mathematical minimums, and for p50 an engineering minimum above the mathematical one. They are a **non-degeneracy gate** only — they do not by themselves guarantee that a calibrated threshold exists for the test that will run. That second condition is given by the existence gate below (§12.5.2.1). Like the existence check, the gate is decided on the actual number of successful latencies after the run; before the run it is a planning check on the expected number (§12.5.3). The gate applies where the decision statistic is the empirical percentile — a baseline-derived assertion, and an advisory raw comparison, where it only marks the result indicative. It does not apply to an enforced explicit requirement, which decides on the count of latencies within the threshold and has its own exact feasibility condition (§12.3.4): at p50 and $\alpha = 0.10$, four latencies all within the threshold demonstrate compliance ($P(Y \geq 4) = 0.0625$), although four is below the p50 minimum.

12.5.2.1 Existence of the Precedence Rank A non-degenerate empirical percentile is necessary but **not sufficient** for the precedence rank of §12.4.2 to exist. The existence condition is the one §12.4.2 states: some rank $k \leq n_b$ must achieve breach$(k) \leq \alpha$ for the test that will run. Because breach(k) decreases in k, a rank exists exactly when the top rank does, breach$(n_b) \leq \alpha$. The condition depends on the baseline size, the test size, the percentile and α together. The test size that decides is the actual number of successful latencies, known after the run; before the run the search on the expected number serves planning (§12.5.3). At p95 and $\alpha = 0.05$, a baseline of 100 latencies cannot support a threshold for a test of 15, and a baseline of 1000 can (rank 998).

The Wilks (1941) table of minimum sample sizes that stood here, $n_s \geq \lceil \log \alpha / \log p \rceil$, is withdrawn: it was the existence condition of the withdrawn confidence bound of §12.4.3, and it is neither necessary nor sufficient for the precedence rank. The same number returns in another role: for an explicit requirement it is the feasibility minimum of `latency/compliance-exact-binomial` (§12.3.4), the fewest successful latencies at which the all-within outcome passes.

For planning, the smallest baseline, no smaller than the test, from which a rank exists at $\alpha = 0.05$ is:

Percentile	$n_t = 10$	$n_t = 25$	$n_t = 50$	$n_t = 100$	$n_t = 200$
p90	33	42	50	100	200
p95	190	86	84	100	200
p99	190	475	950	346	342

The requirement is not monotone in the test size, because the test's nearest rank r moves in integer steps: a p99 test of 50 has $r = 50$, its own maximum, so the baseline must make even the test's largest value unlikely to exceed the threshold. Where $r = n_t$ the breach probability at the top rank is exactly $n_t/(n_b + n_t)$, so a test of 10, 25 or 50 at its maximum needs a baseline of exactly 190, 475 or 950 at $\alpha = 0.05$ — boundaries where the breach probability equals α and the exact-boundary convention of §10.6 decides. At p50 a baseline as large as the test suffices at every test size in the table.

The two percentile gates therefore play different roles for a baseline-derived assertion (an explicit requirement uses neither, §12.5.3):

Gate	Question answered	Outcome
Non-degeneracy (§12.5.2)	Is the empirical percentile distinct from the sample maximum / minimum?	Before the run, on the expected count: a warning and a planning figure. After the run, on the actual count: INCONCLUSIVE for a baseline-derived assertion under VERIFICATION, enforced or advisory; indicative under SMOKE (§12.5.4)
Existence (§12.4.2)	Does some baseline rank achieve a no-degradation breach probability of at most α for this test size?	Before the run, on the expected count: a warning and a planning figure. After the run, on the actual count: INCONCLUSIVE with `saturated: true` when no rank exists

A percentile estimate can be non-degenerate yet still unable to support a calibrated threshold. For each baseline-derived assertion, the framework runs both checks on the expected count before any sample executes, as warnings, and decides both on the actual count after the run.

Scope note on p99.9 and beyond: The supported percentile levels are $\{0.50, 0.90, 0.95, 0.99\}$. Extreme-tail percentiles such as p99.9 are out of scope for the current methodology: a non-degenerate p99.9 estimate requires $n_s \geq 1{,}000$ successful samples, and a statistically useful threshold at 95% requires considerably more. Services with genuine p99.9 SLAs generally warrant dedicated tail-focused instrumentation (production telemetry, HdrHistogram-style log-linear bucketing, or extreme-value modelling) rather than per-test-run estimation.

12.5.3 The Feasibility Gate The checks a latency assertion is subject to depend on its threshold source, because the two rules decide on different statistics: `latency/precedence` on the test's empirical percentile against a baseline order statistic, `latency/compliance-exact-binomial` on the count of latencies within an explicit threshold.

Threshold source	Before the run	After the run, on the actual count
Explicit (§12.3.4)	Exact-binomial feasibility: VERIFICATION refuses a plan below $\lceil \log \alpha / \log p_j \rceil$ (`COMPLIANCE_INFEASIBLE`)	INCONCLUSIVE if no count of the successful latencies can pass
Baseline-derived (§12.4.2)	Test size (`TEST_LARGER_THAN_BASELINE`); non-degeneracy and precedence existence as planning warnings	INCONCLUSIVE below the non-degeneracy minimum (VERIFICATION) or when no rank exists (`saturated: true`)

The mode of the latency dimension (§12.6) changes none of these checks: an advisory assertion is decided by the same rule, so it meets the same refusals before the run and the same INCONCLUSIVE outcomes after it. A refusal still stops the test; an advisory outcome is reported and cannot fail it.

Test size (§5.7.1). Whatever the intent, a test that consumes a baseline is refused when its planned sample size N_t exceeds the sample size N_b of the baseline run (`TEST_LARGER_THAN_BASELINE`). This is the same check, made once on the two samplings, that applies to the test's pass-rate criteria; the numbers of successful latencies, n_t and n_b, are not compared.

For **VERIFICATION** intent, enforced or advisory, the framework then checks before any samples execute:

1. **Non-degeneracy, as a planning check** (§12.5.2), for each baseline-derived assertion: the expected number of successful test samples against the minimum of §12.5.2. When it falls short, the framework warns before the run and reports the planning figure — the smallest planned sample size whose expected count reaches the minimum. The warning is not a refusal: the test runs.
2. **Existence, as a planning check** (§12.5.2.1), for each baseline-derived assertion: the precedence rank search of §12.4.2, run on the baseline's n_b latencies and the test's expected successful-sample count. When it finds no rank, the framework warns before the run and reports the planning figure — the smallest baseline, no smaller than the expected count, that supports a rank for it. The warning is not a verdict: the test runs.
3. **Feasibility of an explicit requirement** (§12.3.4), for each explicit assertion, and the only check it is subject to: the planned number of samples reaches $\lceil \log \alpha / \log p_j \rceil$; otherwise no count could pass even if every sample succeeded, and the design is refused (`COMPLIANCE_INFEASIBLE`). The expected count may be compared with the same minimum for a planning warning; it decides nothing. The percentile non-degeneracy minimum does not apply.

The expected successful-sample count is

$$n_{t,\text{expected}} = \lfloor n_{\text{planned}} \times \hat{p}_{\text{baseline}} \rfloor,$$

with $\hat{p}_{\text{baseline}}$ the baseline run's passing rate, the fraction of its samples that passed every functional criterion (§12.2.1).

The expected count is not a lower bound: a run can return fewer or more successful samples. After the run, both decisions for a baseline-derived assertion are made on the test's actual number of successful samples: below the non-degeneracy minimum the assertion is INCONCLUSIVE under VERIFICATION, enforced or advisory (under SMOKE intent the percentile is evaluated and marked indicative, §12.5.4); otherwise the threshold is derived from the actual count, and the binding saturation decision is made on it (§12.4.2).

Example (non-degeneracy): A p99 assertion with $n_{\text{planned}} = 110$ and a baseline passing rate $\hat{p} = 0.80$ expects 88 successful samples, below the p99 minimum of 100. The framework warns before the run and reports the planning figure: 125 planned samples give an expected 100. The warning binds in neither direction: a run that returns 100 successful latencies is decided, and one that returns 99 is INCONCLUSIVE.

Example: A test with $n_{\text{planned}} = 200$ and a baseline passing rate $\hat{p} = 0.80$ expects $n_{t,\text{expected}} = 160$ successful samples. A p99 assertion at $\alpha = 0.05$ has $r = 159$; against a baseline of 400 latencies no rank achieves α at 160, so the framework warns before the test runs and reports the planning figure: a baseline of at least 554 latencies supports a threshold for 160 (rank 554 at exactly 554). The pre-run result binds in neither direction: against a baseline of 554, a run that returns 161 or more successful latencies finds no rank, and the assertion is INCONCLUSIVE with `saturated: true`.

12.5.4 Indicative Results When the actual number of successful latencies falls below the minimum under SMOKE intent, the framework still evaluates the percentile but marks the result as **indicative**:

> "The p99 result is based on $n_s = 40$ samples (minimum recommended: 100). This result is indicative — a directional signal, not a statistically reliable estimate."

This mirrors the SMOKE/VERIFICATION asymmetry for pass-rate testing (Section 5.7.3). The percentile is computed and reported, but its epistemic weight is explicitly qualified. Indicative marking follows the intent, not the mode: under VERIFICATION an advisory assertion below the minimum is INCONCLUSIVE by its rule, exactly as an enforced one is, and the outcome is reported without failing the test (§12.6).

12.6 Enforcement Modes

Every assertion a test declares is **enforced** by default: its functional criteria and its latency constraints, whether their thresholds are normative (a requirement, §3.6, §12.3.4) or empirical (derived from a baseline, §3.4, §12.4.2). There is no exception by threshold source. An enforced dimension's verdict enters the test verdict V_{test} (§12.3.2), and its FAIL fails the test.

One set of declared assertions for every environment. A test states what the service must do, and the same statement holds wherever the test runs; the method-

ology does not let an assertion count in one environment and not in another. What differs between environments is the evidence an assertion is judged against, and the developer or operator chooses it:

- **Covariates are the developer's declaration.** Declaring covariates (§8.4.1) states that the service's behaviour may depend on the conditions they name; declaring none is itself a declaration that the service's behaviour does not depend on the environment it runs in. Either way the declaration is the developer's, and the matching discipline of §8.4.1 and §8.4.5 applies to it unchanged.
- **An empirical threshold consumes the baseline appropriate to where the test runs.** A baseline measures the service under the conditions of its measure run (§1.5.3). In development the test consumes the developer's own baseline, measured there; in production, a baseline measured in production. Choosing the baseline is the developer's or the operator's; the methodology adds no resolution machinery to choose it for them.
- **A normative threshold refers to its target environment.** A requirement such as ”p95 at most 500 ms” or ”at least 95% of responses well formed” states what the service must achieve where it is meant to run. Where the development environment is significantly slower than production, a latency requirement cannot be met there by a service that meets it in production, and it is incumbent on the developer to switch latency enforcement off in development, as below.

The run-time switch. Each mavai implementation provides a run-time switch — a system property, an environment variable or a command-line flag, in the implementation's own idiom — that makes the **functional** dimension, the **latency** dimension or **both** advisory for a run. The switch acts per dimension, on every assertion of that dimension in the run; it is never a per-test or per-criterion annotation, because the declared assertions are the ones that count. The implementations provide no environment switch: developers and operators know which environment they are in, and choose the switch and the baseline accordingly.

Mode	Decided by	Verdict reported	Enters V_{test}	Can fail the test	Type-I envelopes (§1.4.6)
Enforced	The rule for each criterion or constraint (§3.4, §3.6, §12.3.3)	PASS, FAIL or INCONCLUSIVE	Yes	Yes	Contributes
Advisory	The same rule	PASS, FAIL or INCONCLUSIVE, labelled advisory	No	No	Contributes nothing

What advisory means. An advisory assertion is decided by its proper rule, exactly as an enforced one: a criterion by `regression/fisher` or `compliance/exact-binomial`,

an explicit latency requirement by `latency/compliance-exact-binomial`, a baseline-derived constraint by `latency/precedence`, with the same gates and the same INCONCLUSIVE outcomes (§12.5.3). Its PASS, FAIL or INCONCLUSIVE is reported beside the test verdict and labelled advisory, but it never fails the test. Because none of its decisions can fail the test, an advisory dimension adds nothing to the Type-I envelopes. The settings follow:

Switch setting	V_{test} composes	Reported beside it, advisory
None (default)	V_{rate} and V_{latency}	—
Functional	V_{latency}	V_{rate} and its criteria
Latency	V_{rate}	V_{latency} and its constraints
Both	No dimension: PASS, stated as bound by none	Both dimensions

Configuration refusals still apply. A design that cannot be decided is refused whatever binds: `TEST_LARGER_THAN_BASELINE` and, under VERIFICATION, `COMPLIANCE_INFEASIBLE` (§5.7.1, §12.5.3) refuse an advisory dimension's assertions as they refuse an enforced one's. Under both settings a test therefore cannot fail on its assertions, but it still errors on a refusal or a defect.

Intent is a separate matter. The switch does not change the test's intent. A SMOKE-intent run (§5.7.2) keeps its own treatment: its normative pass-rate and explicit latency verdicts are decided by the SMOKE rules (§5.7.3, §12.3.4) and carry no verification claim, an undersized percentile is marked indicative (§12.5.4), and the raw percentile comparison of §12.3.4 stays available as a labelled figure.

Why enforcement is the default. In methodology 1.5.0 latency assertions were advisory by default and enforcement was opted into, on the ground that a baseline recorded on dedicated CI hardware — $Q_{0.95} = 480$ms, say — would fail a developer's laptop that observes $Q_{0.95} = 650$ms under competing load. That situation is a choice of evidence, not a reason for an assertion not to count. If the threshold is empirical, the laptop's test consumes a baseline measured on the laptop, and a FAIL against it is a genuine regression. If the threshold is normative, it states what production must achieve, and the developer whose laptop cannot meet it switches latency to advisory there and reads the verdict the rule still reaches. A default that made one dimension count only on request would let a declared assertion decide nothing without anyone choosing that it should.

12.7 Relationship to Pass-Rate Testing

The table below summarises how the two quality dimensions parallel each other in statistical treatment:

Aspect	Pass Rate	Latency
Statistical model	Parametric (binomial)	Non-parametric (empirical percentiles)

Aspect	Pass Rate	Latency
Estimand	Success probability p	Percentile quantiles $Q(p_j)$
Decision rule	Fisher cutoff (regression); exact binomial $k_{\min}$ (compliance)	Precedence rank (regression, §12.4.2); exact binomial $y_{\min}$ on the within-threshold count (explicit requirement, §12.3.4)
Baseline storage	$(\hat{p}, k, n)$	$(t_{(1)}, \ldots, t_{(n_s)}, n_s)$
Feasibility gate	$N_{\min} = \lceil \log \alpha / \log p_{\text{req}} \rceil$ (compliance)	By threshold source (§12.5.3): an explicit requirement, the same exact-binomial feasibility minimum $\lceil \log \alpha / \log p \rceil$; a baseline-derived threshold, the non-degeneracy minimum and the precedence rank search; the same in either mode
Indicative marking	Undersized sample note	Undersized sample note
Enforcement	Enforced by default; advisory by run-time switch (§12.6)	Enforced by default, whatever the threshold source; advisory by run-time switch (§12.6)

The two dimensions are evaluated **separately** and, when both are enforced (§12.6), combined with logical conjunction. They are distinct (orthogonal as quality concerns), not necessarily statistically independent: correctness and latency may covary in practice — difficult prompts can be slower and more likely to fail, infrastructure overload can lift both error and tail-latency rates. **The methodology does not require functional and temporal stochasticity to be statistically independent; the combined verdict is a logical conjunction rather than a probabilistic independence model.** What separation guarantees operationally is that latency analysis cannot compensate for a pass-rate failure, and vice versa — each dimension must meet its own threshold.

12.8 Numerical Conventions

The order-statistic constructions of §§12.2 and 12.4 commit to a small set of numerical conventions. They are part of the methodology rather than implementation choices: a downstream implementation that diverges from them produces values that disagree with the mavai-R fixtures.

Sort order. Observed successful latencies are sorted in **ascending** order before any rank lookup. The order statistic $t_{(k)}$ is the k-th smallest value (1-indexed).

Nearest-rank percentile. Per §12.2.2, the empirical percentile is $Q(p_j) = t_{(\lceil p_j \cdot n_s \rceil)}$. This is the same definition the precedence rank of §12.4.2 assumes for the test's percentile; the methodology uses this nearest-rank estimator rather than an interpolat-

ing quantile estimator (e.g. R's Type 7), and the conformance fixtures encode it. The ceiling is taken on the raw product $p_j \cdot n_s$ without intermediate rounding.

Tie policy at integer-millisecond resolution. Wall-clock latencies reported in integer milliseconds induce ties. When a rank lookup $t_{(k)}$ falls inside a run of equal-valued observations, the **largest tied position** is used for threshold rank lookups — the precedence threshold of §12.4.2 and any operator-side query for an upper bound on $Q(p_j)$. This is the **conservative** tie convention: ties at the boundary of a threshold rank do not let the threshold collapse below the value the tied observations actually exhibit, so the bound is not anti-conservatively tight. Point-estimate percentile lookups (§12.2.2) use the rank as written; the conservative tie rule applies specifically to the upper-bound rank k_j of §12.4.2 and to any other order-statistic lookup whose role is to bound a quantile from above.

Displayed-threshold rounding. When a derived latency threshold is rendered for human consumption it is rounded **half-to-even** (banker's rounding) at the displayed precision — typically integer milliseconds for latency. The **raw** values — order statistics, ranks, and any unrounded intermediate quantity — are retained unrounded in the trial record and the transparent-statistics output (§7.1, §10.2), so a downstream consumer can recompute the displayed value from the raw inputs and verify against the mavai-R fixtures without depending on the renderer's precision.

Sort stability. Where the implementation language offers a choice of sort algorithms, the methodology assumes a **stable** sort. Stability has no statistical effect on the order-statistic values, but it makes the trial-record-to-rank mapping reproducible across runs when ties are present, which simplifies post-hoc audit.

The transparent-statistics report (§7.1, §10.2) names the actual sort algorithm and tie convention used at runtime; see the `sortStable` and `tiePolicy` fields specified there.

Appendix A. Elements of the statistical model

The statistical model described in this companion is built from a set of named primitives, parameterised by named statistical quantities, and produces a set of named statistical results. This appendix gathers them in one place, with references to the sections in which each is defined.

The list is restricted to elements that are intrinsic to the model. Operationalization artefacts that frameworks and projects assemble around the model — the service contract as a programming artefact, contract families, binding policies, runtime-resolution mechanisms — are documented elsewhere.

Element	Information content	Defined in
Postcondition	A predicate over the service's output; defines per-trial pass or fail for a single observable property.	§1.4.2

Element	Information content	Defined in
Criterion	The partition unit of the functional dimension. References one or more postconditions, has a mode (inferential or observational), and where inferential carries an error (significance) level α_c and a threshold: a normative requirement p_{req} (compliance), or a reference to a baseline whose counts give the integer cutoff c at resolution time (empirical regression). Its denominator is the full sampling N (§1.4.5a).	§1.4.2, §1.4.3, §1.4.5, §1.4.5a
Sampling	A list of $N \geq 1$ samples posted to the service under test in a single experiment. Shared by every criterion of the experiment; per-criterion verdicts are claims about the same N samples.	§1.4.2, §1.4.7
Factor record	The identification of the service, model, and serving configuration whose p_c is being estimated. Two evaluations that differ in factors evaluate two different objects.	§1.3.1
Covariate profile	The recorded values of declared contextual variables at the time of an evaluation; affects baseline comparability.	§8.4.1
Inferential reach of the sampling	The N samples the experiment posts to the service, over which every criterion in the experiment is exercised; a criterion's verdict or evidence is, primarily, a claim about those samples. Extending it to a different input distribution is a separate interpretive move (§8.4.6).	§1.4.7
Per-criterion Bernoulli stream	The sequence of per-criterion indicators $\{X_{i,c}\}$ treated as i.i.d. Bernoulli with parameter p_c under the model's working approximation.	§1.4.3

Element	Information content	Defined in
Effective denominator (n_c)	The sampling size N: every sample is seen by every criterion. A trial is a success only on a clean PASS; every other outcome — including a transform/no-value failure — is a counted FAIL, never excluded. `zeroFailures` (observational) criteria carry no rate denominator.	§1.4.5a
Confidence statement	A one-line statement of what the per-criterion error level controls, naming the random experiment (§7.2); descriptive intervals use Wilson (§2.3.1).	§7.2, §2.3.1
Decision rule	The versioned procedure that decides a criterion: `regression/fisher` v1, `compliance/exact-binomial` v1, `latency/precedence` v1 or `latency/compliance-exact-binomial` v1, each with its epistemic status.	§3.4, §3.6, §12.4.2, §9
Regression cutoff (c)	The binding decision artefact of empirical regression: the smallest test count whose one-sided Fisher p-value exceeds α, from (K_b, n_b, n_t, α). PASS iff $K_t \geq c$.	§3.4
Compliance count ($k_{\min}$)	The binding decision artefact of normative compliance: the smallest count the exact one-sided binomial test accepts. PASS iff $K \geq k_{\min}$.	§3.6
Latency rank	The binding decision artefact of latency regression: the precedence rank k and threshold $t_{(k)}$, or `saturated` (INCONCLUSIVE).	§12.4.2
Latency compliance count	The binding decision artefact of an explicit latency requirement: the smallest passing count $y_{\min}$ of successful latencies at or below τ; PASS iff $Y \geq y_{\min}$, INCONCLUSIVE when too few successful latencies arrived for any count to pass.	§12.3.4
Design alternative rate	The true rate p_{design} at which a regression test must reach its target power. Not a tolerance: the test flags any degradation from the baseline. The design power and the resolved-test power are stated at it, named apart.	§5.3, §5.4.1

Element	Information content	Defined in
Configuration error	A named refusal before any sample runs: `TEST_LARGER_THAN_BASELINE`, `COMPLIANCE_INFEASIBLE`; every applicable code is reported, in that order.	§5.7.1
Threshold origin	The provenance category of an inferential threshold (SLA, SLO, POLICY, EMPIRICAL, UNSPECIFIED), recorded with the requirement or, for EMPIRICAL, the baseline reference.	§7.4
Sample-size requirement	The per-criterion sample count from which the exact power of the operative rule stays at its target, with the feasibility gate that refuses a compliance design too small to pass.	§§5.4–5.5, §5.7.1
Per-criterion verdict	PASS, FAIL, or INCONCLUSIVE on a criterion: for **compliance** criteria, the observed count's relation to $k_{\min}$; for **regression** criteria, the observed success count's relation to the Fisher cutoff c_c derived from the baseline's counts at α_c; for **observational** criteria, the zero-failure observation. Carries the supporting statistics, the threshold and origin, α_c, and the population specification.	§1.4.3, §1.4.5, §1.4.6
Composite verdict	A structured tuple over per-criterion verdicts; its structural composite is the functional verdict V_{rate}.	§1.4.6
Latency verdict	V_{latency}, the structural composite of a test's latency constraints, each decided by its rule: PASS if every one passes, FAIL if any fails, INCONCLUSIVE otherwise. Formed the same way whether the latency dimension is enforced or advisory; an advisory one is reported beside the test verdict and does not enter it.	§12.3.2

Element	Information content	Defined in
Test verdict	V_{test}, the top-level verdict of a test: the structural composite of its enforced dimensions among V_{rate} and V_{latency} — PASS if every enforced dimension passes, FAIL if any fails, INCONCLUSIVE otherwise; V_{latency} alone when the test has no functional criteria or its functional dimension is advisory, V_{rate} alone when it asserts no latency constraint or its latency dimension is advisory, and PASS, bound by no dimension, when both are advisory. The report names the criteria and latency constraints that decided it.	§12.3.2, §10.2
Enforcement mode	Enforced or advisory, per dimension (functional, latency) and for a whole run: enforced by default for every assertion whatever its threshold source; advisory by a run-time switch, never by a per-test annotation or an environment setting. An advisory dimension is decided by its rules and reported, enters neither V_{test} nor the Type-I envelopes, and stays subject to the configuration refusals.	§12.6
Composite Type-I envelopes	Procedure-direction-specific union-bound aggregates over inferential criteria: the **false-degradation-signal envelope** $\alpha_{\text{fds}} \leq \sum_{c \in \text{regression}} \alpha_c$ and the **false-compliance envelope** $\alpha_{\text{fc}} \leq \sum_{c \in \text{compliance}} \alpha_c$. For a whole test they also include the latency decisions: baseline-derived latency constraints in the false-degradation-signal envelope, explicit latency requirements in the false-compliance envelope (§12.3.1). A test reports each envelope only when it makes binding decisions of that direction. Observational criteria and an advisory dimension's criteria and constraints (§12.6) contribute to neither.	§1.4.6

Element	Information content	Defined in
Baseline	A family of per-criterion point estimators $\{\widehat{p}_c\}$ with supporting $\{n_c\}$ and $\{K_c\}$, qualified by the factor record, covariate profile, structural reference, and optional expiration window under which it was measured. Consumed by inferential criteria of origin EMPIRICAL, whose counts derive the cutoff c_c at resolution time.	§1.5

References

1. Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association, 22(158), 209-212.
2. Agresti, A., & Coull, B. A. (1998). Approximate is better than ”exact” for interval estimation of binomial proportions. The American Statistician, 52(2), 119-126.
3. Brown, L. D., Cai, T. T., & DasGupta, A. (2001). Interval estimation for a binomial proportion. Statistical Science, 16(2), 101-133.
4. Newcombe, R. G. (1998). Two-sided confidence intervals for the single proportion: comparison of seven methods. Statistics in Medicine, 17(8), 857-872.
5. Hanley, J. A., & Lippman-Hand, A. (1983). If nothing goes wrong, is everything all right? Interpreting zero numerators. JAMA, 249(13), 1743-1745. [The ”Rule of Three”]
6. Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika, 26(4), 404-413.
7. Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). Bayesian Data Analysis (3rd ed.). Chapman and Hall/CRC. [Beta-Binomial posterior predictive]
8. Jeffreys, H. (1946). An invariant form for the prior probability in estimation problems. Proceedings of the Royal Society of London. Series A, 186(1007), 453-461. [Jeffreys prior]
9. Hyndman, R. J., & Fan, Y. (1996). Sample quantiles in statistical packages. The American Statistician, 50(4), 361-365. [Percentile interpolation methods]
10. David, H. A., & Nagaraja, H. N. (2003). Order Statistics (3rd ed.). Wiley-Interscience. [Order statistics and empirical percentiles]
11. Conover, W. J. (1999). Practical Nonparametric Statistics (3rd ed.). Wiley. [Nonparametric quantile confidence intervals via order statistics; the binomial construction of §12.4.2]

12. Hollander, M., & Wolfe, D. A. (1999). Nonparametric Statistical Methods (2nd ed.). Wiley. [Parallel reference for the order-statistic upper bound]

13. Wilks, S. S. (1941). Determination of sample sizes for setting tolerance limits. Annals of Mathematical Statistics, 12(1), 91-96. [Foundational treatment of distribution-free tolerance intervals]

14. Krishnamoorthy, K., & Mathew, T. (2009). Statistical Tolerance Regions: Theory, Applications, and Computation. Wiley. [Modern treatment of distribution-free tolerance and quantile bounds]

15. Bayarri, M. J., & Berger, J. O. (2004). The interplay of Bayesian and frequentist analysis. Statistical Science, 19(1), 58-80. [Framework for reconciling predictive-Bayesian and frequentist inference, cited in §4.5]

16. Geisser, S. (1993). Predictive Inference: An Introduction. Chapman and Hall. [Predictive treatment of binomial future performance]

17. Anthropic (2026). Models and Model deprecations. Claude documentation, accessed 2026-05-11. https://platform.claude.com/docs/en/docs/about-claude/models/overview ; https://platform.claude.com/docs/en/docs/about-claude/model-deprecations [Provider commitment that every Claude model ID is a pinned snapshot for the lifetime of its deprecation window; cited in §1.3.1 for the snapshot-vs-floating-alias distinction.]

18. Chen, L., Zaharia, M., & Zou, J. (2023). How is ChatGPT's behavior changing over time? arXiv preprint arXiv:2307.09009. [Empirical demonstration of behaviour drift in floating-alias model identifiers across snapshots over a multi-month period; cited in §1.3.1 as motivation for the pinned-snapshot prescription.]

19. Liang, K.-Y., & Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. Biometrika, 73(1), 13–22. [Generalised estimating equations; cited in §8.2 / §8.2.1 for clustered-design analysis under non-trivial dependence structures.]

20. Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society: Series B, 57(1), 289–300. [Original BH procedure; cited in §7.3 for FDR control under independence and PRDS.]

21. Benjamini, Y., & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. Annals of Statistics, 29(4), 1165–1188. [BY procedure; cited in §7.3 for FDR control under arbitrary dependence.]

22. Anthropic. Enterprise deployment overview. Claude documentation, accessed 2026-05-14. [Provider documentation describing endpoint and routing configurations that vary across platforms; cited in §1.3.1 for the necessary-but-insufficient status of pinned model IDs.]

23. Garces Arias, E., Blocher, H., Rodemann, J., Aßenmacher, M., & Jansen, C. (2025). Statistical multicriteria evaluation of LLM-generated text. In Proceedings of the 18th International Natural Language Generation Conference

(INLG), 338–351. Association for Computational Linguistics. [Independent diagnosis of the multi-criterion evaluation problem for stochastic language systems, addressed via a Generalized Stochastic Dominance front rather than the per-criterion decomposition adopted here; cited in the Introduction.]

24. Fisher, R. A. (1935). The logic of inductive inference. Journal of the Royal Statistical Society, 98(1), 39–82. [The exact test for a 2 × 2 table conditional on its margins; the basis of the regression rule of §3.4.]

25. Nelson, L. S. (1963). Tables for a precedence life test. Technometrics, 5(4), 491–499. [Precedence tests comparing two samples through order statistics; cited in §12.4.2.]

26. Balakrishnan, N., & Ng, H. K. T. (2006). Precedence-Type Tests and Applications. Wiley. [Modern treatment of precedence tests; cited in §12.4.2.]

27. Berger, R. L. (1982). Multiparameter hypothesis testing and acceptance sampling. Technometrics, 24(4), 295–300. [Intersection–union tests: a claim that requires every component test to reject is a level-α test when each component is; cited in §1.4.5b.]

This document is intended for review by professional statisticians. For operational guidance, see the documentation in your framework of choice: punit, feotest. For the reference implementation of all statistical computations described here, see mavai-R.