

Human-Governed Validation of R Packages for Regulated Statistical Computing: LLM-Assisted Specifications and Auditable Test Evidence

[Author name]*

Abstract

Open-source R packages are increasingly used in regulated statistical computing, but risk scores, coverage, and upstream tests do not by themselves demonstrate fitness for an intended use. We present a specification-driven prototype developed as independent research. Versioned specifications define requirements and acceptance criteria; an isolated engine combines known-answer, metamorphic, property-based, fuzz, and mutation tests; and records can enter a tamper-evident hash chain. An optional local large language model drafts structure only: expected values are captured characterization baselines or independently derived, and approval remains human. A public contract replay passed 15/15 LLM authority-boundary checks, including a prompt-injection red-team set. Three numerical capsules evaluate the test design. A pristine synthetic R package passed 11/11 cases spanning four test techniques, while four pre-injected defect variants each failed the targeted requirement. A jsonlite replay showed that the documented `digits=4` default uses at most four decimal places rather than four significant digits; maximum precision requires `digits=NA`. A synthetic time-to-event capsule reproduced prespecified Kaplan–Meier and Cox-model outputs and passed 50/50 row-order invariance checks. Mutation process failures are classified as invalid/stillborn rather than detections, explicit source files define the sampling frame, and surviving mutants remain equivalence-undetermined. The public artifacts include source, specifications, machine-readable outputs, replay scripts, and SHA-256 manifests. These results demonstrate test-oracle construction, fail-closed model governance, and auditable correction; they do not establish package quality, regulatory compliance, LLM superiority, or general fault-detection probability.

Keywords: R package validation; large language models; metamorphic testing; mutation testing; computerized system validation; audit trail

1 Introduction

Open-source statistical software is increasingly important in drug development, making software reliability, sustainability, and traceable use part of the statistical operating model.^{1, 2} Current R package-validation practice emphasizes risk-based screening using evidence on testing, documentation, maintenance, community use, and licensing.^{3, 4, 5} Such screening supports prioritization, but does not by itself answer three intended-use evidence questions:

1. **What did you require?** — a versioned statement of requirements with acceptance criteria, specific to how *you* use the package;

*Independent researcher. Corresponding-author identity, ORCID, and contact details will be provided in the submission system.

2. **How do you know the package meets it?** — executed evidence linking each requirement to concrete test cases and results;
3. **How do you know your tests would detect a relevant fault?** — evidence about the test suite itself, not only the package.

Mutation analysis is one established way to examine test-suite adequacy, but its interpretation depends on the mutation operators and on the treatment of equivalent or invalid mutants.^{6, 7} The test-oracle problem is a separate obstacle when exact expected results are unavailable or costly to obtain.⁸ Against this background, we describe an implemented research prototype. Because specification authoring is itself a scaling bottleneck, the pipeline also includes an optional local large language model (LLM) drafting path that cannot create an expected result, approve a specification, or issue a validation verdict. The work makes four contributions:

1. **A metamorphic-oracle framework for R packages** (Section 3.4). The no-oracle problem — you cannot know the expected output for arbitrary inputs — is the core obstacle to testing beyond a package’s own suite. We apply metamorphic testing^{9, 10, 11} through a fixed, auditable relation library (roundtrip identity, vector and row permutation invariance, scale covariance, subset consistency) driven by deterministic seeded generators. In public replay, a round-trip relation identified a documented numeric-precision default that did not meet the replay’s declared intended use (Section 4.3).
2. **Operator- and scope-conditioned mutation analysis with zero external dependencies** (Section 3.5). We characterize each spec suite against self-injected source mutation — token-level mutants produced from R’s own parse data, installed and tested in child R processes, without third-party mutation tooling — under a *green precondition* that refuses to report a score when the suite fails on the unmodified package. Explicit source files define the intended-use sampling frame; process failures and undecidable mutants are not credited as detections. Public canary evidence and contract tests exercise fail-closed behavior (Section 4.2).
3. **A provenance-separated LLM drafting workflow** (Section 3.2). The model proposes requirements, case structure, calls, tolerances, generators, and seeds from bounded package metadata, while golden-master expected values remain null. Those fields can be populated only by sandbox execution — which yields candidate characterization baselines, not independent evidence — or by human-supplied independently derived values; a mandatory `oracle_source` declaration (`independent_derivation`, `characterization_capture`, or `second_implementation`) records which. Strict schema validation, promotion, and human electronic-signature approval remain outside model authority. Model identifier, complete prompt, drafts, hashes, repair errors, and lifecycle transitions are retained.
4. **An end-to-end integrity and human-accountability loop** (Section 3.6): tamper-evident hash-chained audit records, electronic-signature prototype controls informed by selected 21 CFR Part 11 principles,^{12, 13} per-item human review records bound by hash to the verdicts they dispose of, and a package repository that exposes only approved artifacts. Automated processes do not sign; review disposition requires a named human. The prototype demonstrates the technical binding and fail-closed states; it does not claim a completed regulated approval or Part 11 compliance determination.

Section 4 uses only public synthetic or CRAN-package replays executed in September 2026. The packages and examples were selected to exercise mechanisms rather than represent CRAN; machine-readable evidence, source, and executable replay scripts are supplied as supplementary material.

2 Background: What Screening Evidence Does Not Show

Regulatory frame. Risk-based life-cycle models such as GAMP 5 inform computerized-system assurance in regulated life sciences,¹⁴ while 21 CFR Part 11 applies when electronic records or signatures fall within its scope.^{12, 13} FDA’s general software-validation guidance also emphasizes intended use, life-cycle controls, and traceability, although its stated scope is medical-device software and software used to design or manufacture devices.¹⁵ For R, the R Foundation guidance summarizes development and testing controls for the language,² and the R Validation Hub white paper proposes risk-based assessment of contributed packages.³ PHUSE guidance describes supporting controls such as lock files, snapshot repositories, and change-controlled environments.⁵ These sources motivate intended-use testing but do not prescribe a mutation- or metamorphic-testing implementation.

Prototype scope. The implementation separates screening-grade package risk assessment from deeper intended-use validation. This paper addresses only the latter. Its hardening rules are method-driven: failed, missing, or unreviewed evidence downgrades the conclusion; classification and sign-off are reserved for humans; and report claims are either linked to executed evidence or suppressed. No vendor comparison or organizational deployment claim is made.

3 Methods

3.1 Architecture: a gateway that never runs untrusted code

The pipeline has two halves that communicate only through artifact files on a shared volume (Figure 1):

- **The gateway** owns specifications, reports, the audit chain, and the gated repository, but never executes assessed-package code.
- **The sandbox** installs and checks the package and runs its tests and the validation engine. Requests and results are atomically written JSON artifacts, so readers do not observe partial records.

Smoke-stage results (installation, R CMD check, upstream tests, and coverage) remain separate from intended-use validation. Neither can mask the other: a passing intended-use suite alongside a failed smoke stage is a non-clean result requiring review, not an unqualified package pass.

3.2 LLM-assisted drafting without LLM-authored evidence

Hand-authoring specifications is a scaling bottleneck, but a model-authored oracle would make plausible text indistinguishable from evidence. The optional local, Ollama-compatible path therefore separates drafting from every authoritative decision (Figure 2).

The prompt contains fixed rules, an exemplar, and only the tarball’s DESCRIPTION and exported NAMESPACE symbols. The LLM may propose intended use, requirements, calls, tolerances, generators, and seeds, but every fresh golden-master **expected** field must be null. The prompt, model identifier, raw draft, hashes, and validation errors are retained. Restricted-YAML parsing and schema checks permit one repair attempt; an invalid draft cannot advance.

In separate **capture** mode, the sandbox executes proposed calls against the pinned package and records the returned value as a *candidate*: a characterization baseline of current package behavior, not independent evidence, because a value captured from the package under test is a circular oracle for that package’s correctness. A value is captured only for a successful, lossless, JSON-safe scalar

Two-half validation architecture: artifact-only communication, hash-chained evidence, human-only signatures

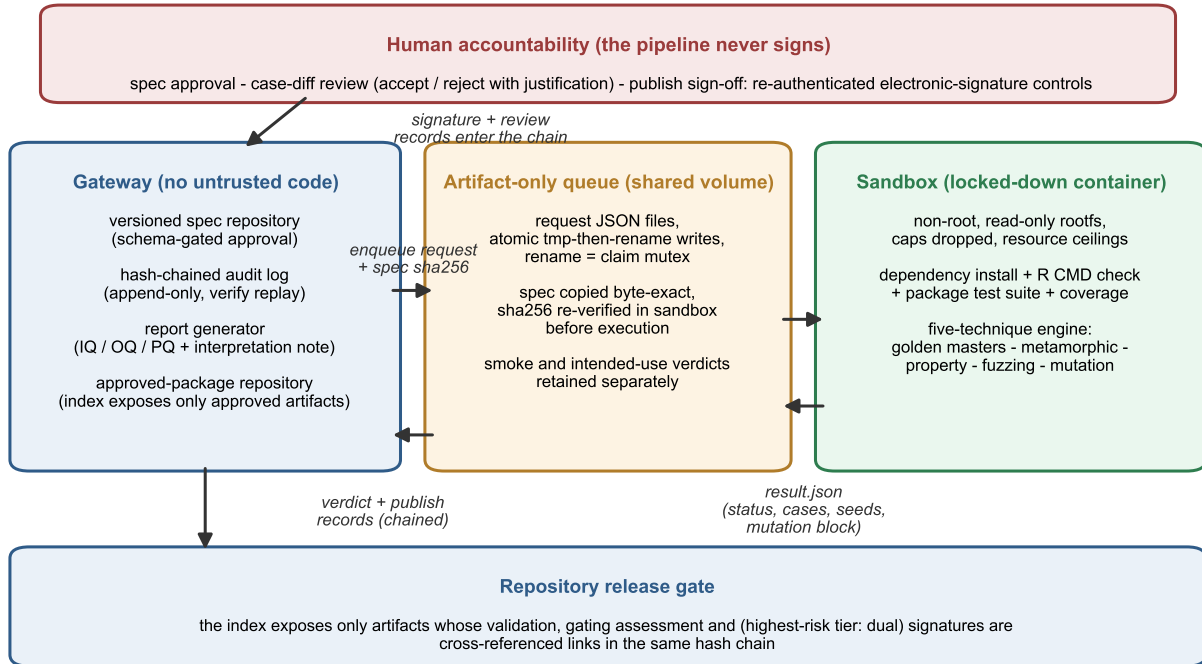


Figure 1: Two-half validation architecture. The gateway (audit chain, specs, reports, approved repository) never executes assessed-package code; the sandboxed engine does, receiving requests and returning results exclusively via atomically written artifact files. Human electronic-signature records (21 CFR Part 11-style) and review decisions enter the same hash chain as every automated verdict; the repository serves only packages whose verdict, gating assessment, and signatures are cross-referenced chain links.

or array; otherwise the field stays null with a note. Every populated case must declare its provenance in a mandatory `oracle_source` field whose enumerated values — `independent_derivation`, `characterization_capture`, and `second_implementation` — are enforced by the schema validator. The pre-capture draft is preserved, and strict schema validation gates byte-exact promotion. Human review remains required: a named reviewer signs off that each captured characterization baseline is acceptable for the declared intended use, and this sign-off — not the capture itself — is the operative control against circular-oracle risk. Thus the LLM cannot supply an oracle, classify a discrepancy, sign, approve, publish, or turn a result into pass.

Contract tests covered disable behavior, provenance retention, malformed completion, one-shot repair, endpoint failure, lossless/lossy capture, byte-exact promotion, audit records, and routes. Model and sandbox endpoints were mocked or stubbed; the R capture suite ran separately. Supplementary Material S2 states this verification boundary: implementation controls were tested, not draft quality, time savings, or validation effectiveness.

LLM assistance is separated from executable evidence and release authority

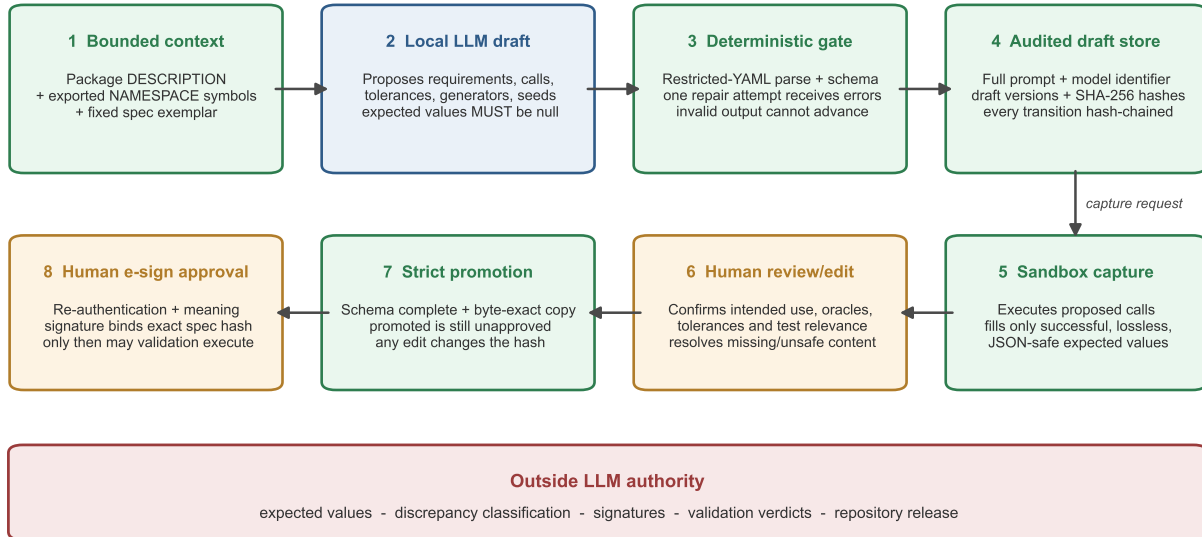


Figure 2: Provenance-separated LLM drafting. The model can propose only specification structure. Executed package values, strict schema gates, human review, signatures, validation verdicts, and release remain outside model authority. Every transition retains the artifacts and hashes needed to reconstruct what the model saw and produced.

3.3 Validation specs: requirements as versioned, signed artifacts

Each package carries a versioned specification declaring intended use, requirements with unique IDs and risk tiers, requirement–case links, and five test sections. A restricted YAML subset is parsed by both gateway and R engine.

Three integrity properties are structural:

- **Schema validity gates approval.** Referential integrity, unique IDs, enum membership, and tolerance sanity are checked before a spec can run.

- **Approval is a signature, not a flag.** Spec approval is an electronic-signature record whose object hash is the spec file’s current SHA-256 digest; editing an approved spec invalidates its approval automatically because the hash no longer matches.
- **Executed evidence is bound.** The gateway copies the approved spec byte-exactly into the run directory; the request records its hash, which the sandbox verifies before execution.

Requirements, expected values, intended-use boundaries, and tolerances remain human responsibilities. Review exposes canary qualification, any eligible scope-conditioned mutation diagnostic, and seeded replay information. Sign-off covers those criteria and evidence, never general detection power.

Requirement rollup is pessimistic: any linked failure or error fails the requirement; a `diff` blocks pass pending review; a missing case leaves the requirement **untested**.

3.4 The five-technique engine

The sandbox records one fixed seed per case and executes five techniques:

1. **Known answers.** Calls are compared with scalar, array, or file values using $|a-b| \leq abs+rel|b|$; value, structure, length, or type mismatches become blocking `diff` records.
2. **Metamorphic relations.** A fixed enum library of round-trip identity ($t_2(t_1(x)) = x$), vector- and row-permutation invariance, scale covariance, and subset consistency is applied to named functions with seeded generators. Failures retain a counterexample hash and summary. These relations provide oracles when exact outputs are unavailable.^{9, 11}
3. **Properties.** Predicates follow the QuickCheck paradigm¹⁸ and are executed as deterministic seeded randomized property checks: the engine’s own fixed-enumeration generators reconstruct the same inputs from the recorded seed on every replay, and each predicate is checked for every generated case. Randomized shrinking is deliberately not used; deterministic, auditable replay is prioritized over automatic counterexample minimization.
4. **Invalid-input fuzz cases.** Seven canonical mutations (missing, infinite, wrong-type, empty, malformed, and oversized values) test a declared graceful-error contract. This deterministic corpus is not coverage-guided fuzzing.²²
5. **Mutation testing of the spec suite itself** — Section 3.5.

Reports map environment fingerprints, requirement–case execution, and intended-use scenarios into an IQ/OQ/PQ skeleton, with an explicit statement that automation does not replace human sign-off.

3.5 Operator-conditioned mutation analysis

Passing tests do not establish sensitivity to code changes. The self-contained mutator uses R parse data within explicitly justified package-relative `source_files`. A recorded seed samples an ordered candidate pool; candidate count, locations, operators, and verdicts are retained. Operators swap arithmetic or relational tokens, perturb constants by one, or negate conditions.

Each one-edit source copy is installed in a fresh library and tested in a child R process. A decided test failure is **detected**; a pass is **survived**. The six failure classes — install, load, runner, engine, malformed-result, and missing-verdict — are **invalid/stillborn**, never detections; wall-clock exhaustion is **timeout**. Survivors remain **equivalence-undetermined** until review. Any descriptive ratio is *detected*/(*detected* + *survived*) among executable sampled mutants only.

A green precondition withholds the ratio (`score = null`) unless the unmodified package first passes. The ratio is conditional on declared scope, operators, seed, and executable mutants; it estimates neither defect rate nor general detection probability. Equivalent mutants can depress it, whereas misclassified crashes inflate it.⁷

3.6 Evidence integrity and the human-on-the-loop

Tamper-evident audit chain. Assessments, validations, reviews, signatures, and publishes form an append-only JSONL chain with previous- and record-hashes over canonical serialization.²³ Validation records bind the spec and verdict; review records bind a justified human disposition to that validation. Tests verify construction, replay, and failure on alteration; the failure-on-alteration checks are publicly replayed against six tamper classes in Supplementary Material S6. The exporter verifies the complete source chain and records both source- and exported-byte hashes, but a selected export is not called a complete-chain replay.

Electronic-signature controls. Re-authentication is mandatory and an unreachable identity service refuses signing. A fixed meaning and content hash bind each signature to the exact object, so editing invalidates approval. These are prototype controls, not a Part 11 compliance determination.

Repository hard gate. The index exposes only approved artifacts; the highest tier requires two distinct signatures linked to the gating and publish records. Automation never signs. Pending fields remain visibly unsigned, and the public evaluation represents only tested fail-closed states and byte bindings, not completed regulated signatures.

3.7 Explicit degradation: no silent failure states

Every path writes one of six fixed verdicts: install failure, resource limit, check error, test failure, engine error, or ok. Coverage failure becomes `percent: null`. Spec-hash mismatch, unreadable results, timeout, or a skipped validation stage creates an explicit error block without altering the smoke-stage result.

3.8 Sandbox security posture

Assessed code runs non-root with a read-only root, dropped capabilities, no-new-privileges, resource ceilings, bounded temporary storage, a pinned repository, and pre-execution tarball hash verification. Repository pinning is only application-layer egress control; a network proxy remains necessary to prevent arbitrary outbound connections.

4 Evaluation

All reported evaluation artifacts are public and contain synthetic data or public CRAN-package behavior. Replays were executed locally on 2026-09-05 with R 4.5.1. Supplementary Material S1 contains the jsonlite and survival replays; S2 documents and executes the LLM governance contract; S3 contains the synthetic canary source, injected variants, specification, and machine-readable results; and S6 replays the tamper-evident audit-chain verification and its six alteration tests on a synthetic chain. A bundle manifest binds every submitted artifact. The evaluation is organized as five research questions (RQ1–RQ5): framework sensitivity, metamorphic oracles, the LLM authority boundary, exact replayability, and mutation-funnel operability. Each subsection states its predefined success criterion first and then reports the result against it.

4.1 Evaluation design

Three public capsules were selected purposively to exercise distinct controls (Table 1): the synthetic canary tests framework sensitivity to documented injected changes; jsonlite 2.0.0 tests a numeric round-trip contract; and survival 3.8-3²⁴ tests a pharmaceutical time-to-event workflow. The survival capsule compares pooled Kaplan–Meier estimates and a single-covariate Cox coefficient, standard error, and partial log-likelihood with prespecified values, then checks row-order invariance over 50 seeded permutations. Results are descriptive and do not compare package quality.

Oracle provenance is explicit in every capsule. A value captured from the package under test is a characterization baseline: it fixes current behavior and, standing alone, is a circular oracle. The jsonlite serialization master (GM-001 in its spec) is therefore declared `oracle_source: characterization_capture` and requires human sign-off before use. Non-circular known-answer evidence is carried by the survival capsule, whose Kaplan–Meier and Cox known answers (GM-001/GM-002) are declared `oracle_source: independent_derivation` — product-limit hand arithmetic and constants from a prespecified analysis — while the metamorphic relations need no expected output at all. The capsules map to the research questions as follows: canarycalm addresses RQ1, jsonlite RQ2, the S2 governance capsule RQ3, the seed-bound replays of all capsules RQ4, and the mutation funnel RQ5.

Table 1: Public evaluation capsules (Figure 3 shows the mutation-accounting funnel that governs adequacy reporting). Mutation fractions are not reported by the portable replays, which disable source-package mutation; the dedicated funnel run for RQ5 is pending (Section 4.6).

Capsule	Purpose	Observed result
canarycalm 1.0.0	framework qualification with public source	pristine 11/11 pass; each of four injected variants fails its targeted requirement
jsonlite 2.0.0	numeric serialization calling contract	exact outputs confirm decimal-place, significant-digit, and maximum-precision semantics
survival 3.8-3	synthetic time-to-event analysis	Kaplan–Meier and Cox known answers pass; row-order relation passes 50/50

4.2 RQ1: Framework sensitivity — targeted, per-requirement detection

Success criterion (predefined). The pristine canary package passes all 11 cases and all four requirement rollups, and each of the four pre-injected defect variants fails exactly its targeted requirement, with no unexpected requirement failing.

Result. The pipeline’s own behavior is a precondition for interpreting package verdicts. The dependency-free {canarycalm} package has four documented functions and 11 cases: six known answers, two metamorphic relations, one property, and two invalid-input fuzz cases. The pristine source passed 11/11 and all four requirement rollups, establishing the green baseline. Each pre-injected source change then produced its prespecified failure pattern, including both a known-answer and property failure for broken NA handling: arithmetic replacement failed GM-001 and GM-002; incorrect scaling failed GM-003; broken NA removal failed GM-004 and PB-001; and broken string formatting failed GM-006. Every variant failed only its targeted requirement and no unexpected requirement failed, so the criterion is met. This is stronger evidence that the four active test techniques execute and roll up correctly than a report based only on a passing package. Supplementary Material S3 includes all source and raw result files. It remains a deliberately small

Corrected mutation accounting: every exclusion remains visible

Raw ratio = detected / (detected + survived), conditional on scope, operators, seed, and executable sample

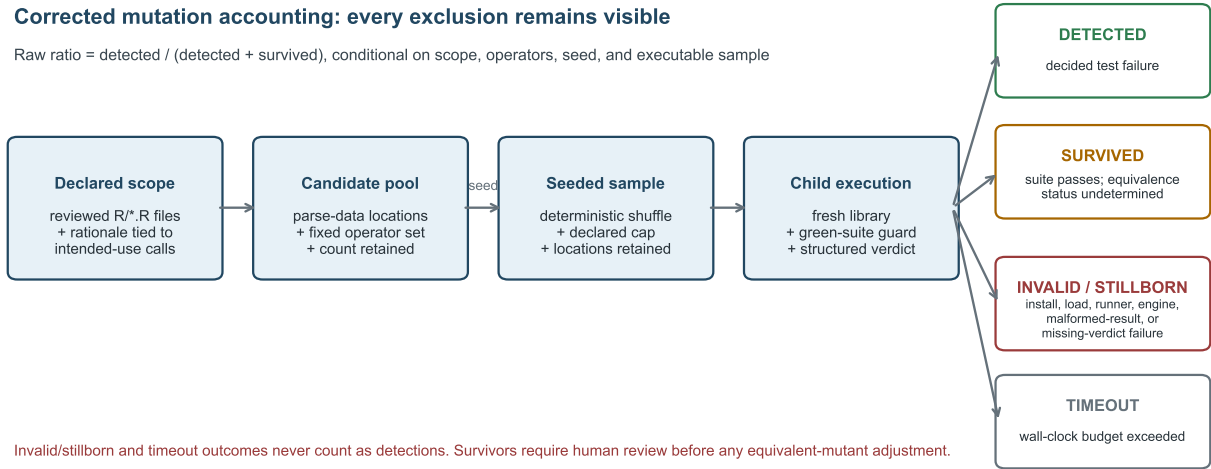


Figure 3: Corrected mutation accounting. A declared intended-use source scope and seeded candidate sample precede execution. Only decided test failures of executable mutants are detected; invalid/stillborn and timeout outcomes remain separate, while survivors remain equivalence-undetermined until review. The raw ratio is conditional on this funnel and is not general defect-detection power.

seeded experiment, not a sensitivity estimate over realistic faults; mutation is disabled in the portable replay and is addressed by the dedicated funnel run (RQ5).

4.3 RQ2: Metamorphic oracles — an oracle-free calling-contract finding

Success criterion (predefined). A seeded metamorphic relation replays publicly and either holds over its declared input domain or yields a specific, reproducible finding with a stated remediation.

Result. jsonlite’s spec anchored a `roundtrip_identity` relation on `toJSON/fromJSON` over seeded numeric vectors. In the public replay (Supplementary Material S1), the seeded relation failed at its first case (seed 43): under the documented default `digits = 4` — which limits the number of decimal digits printed²⁵ — a generated two-element vector serializes as `[-0.0375,-1.5746]` although the generated values carry additional decimal places, without an error or warning. For a fixed example, jsonlite 2.0.0 serializes the length-one vector 82.15321 as `[82.1532]` under the default. This is documented behavior, not a package defect; it did not meet the replay’s declared round-trip tolerance over the generated input domain: all 50 seeded cases violated the default policy, while all 50 passed under `digits = NA`. The criterion is therefore met on the finding branch: the oracle-free relation produced a reproducible calling-contract mismatch. The remediation was a calling-contract change rather than a package patch. The corrected candidate spec v0.2.0 uses `digits = NA` for maximum precision and retains a numerical tolerance; `digits = I(4)` is the distinct significant-digits form. S1 includes the exact input/output record, seeded relation replay, corrected specification, replay code, and hashes. This independent research artifact is not represented as a regulated approval or as evidence of a jsonlite defect.

4.4 RQ3: LLM authority boundary — governance fails closed

Success criterion (predefined). A deterministic public replay through the implementation’s own schema validator rejects every model-authored expected value, authority field, unknown field, deny-listed call, and invented oracle-provenance value; accepts only null-oracle fresh drafts and

lossless captures; and blocks promotion while any expected value is pending — with every check, including a prompt-injection red-team set, passing.

Result. The public contract replay (Supplementary Material S2) passes 15/15 checks. It executes the gateway’s own restricted-YAML parser and spec validator as verbatim, hash-attested capsule modules: fresh drafts keep every golden-master `expected` field null; a model-authored expected value, golden file, authority field (at any nesting level, including Unicode-confusable spellings), unknown field, or invented `oracle_source` value is rejected; deny-listed R calls are rejected in every spec expression field; lossless capture is accepted and lossy capture rejected; and promotion is blocked while an expected value is pending. A prompt-injection red-team replay feeds adversarial completions — what a model would produce if it obeyed an injected instruction — to the same real validator: 10/10 schema-layer injections are rejected. One documented scenario, an invented export with a harmless name, is beyond static schema validation by construction; it is refused at execution by the sandbox as an explicit error case, and an AST-level deny/allow gate on every spec R string is enforced before evaluation, so a validator bypass cannot make a spec string execute a denied call. The implementation also passed 56 gateway spec-drafting tests and a 69-test strict-schema/authority suite in the local verification environment. These tests exercised the authority boundary — especially null expected values before execution, rejection of lossy captured values, strict promotion, and retention of prompt/draft hashes — rather than comparing models or measuring authoring performance.

Real-model drafting experiment. To test the boundary with actual models rather than only contract fixtures, a small preregistered-style run (Supplementary Material S8) asked two frozen local open-weight models (qwen3:8b and qwen3-coder:30b, weight digests, quantization, serving version and per-run seeds recorded) to draft specs for three public R packages (digest, jsonlite, survival — the latter two used directly in pharmaceutical-statistics workflows), three repeats each, through the implementation’s verbatim prompt builder on a CPU-only local server. Of 18 primary runs, 2 produced schema-valid drafts (both the larger model on jsonlite; one on first attempt, one after the single repair retry). No raw completion contained an authority field or a deny-listed call; one contained a hallucinated API call, and the dominant rejection class was the strict whitelist itself (25 unknown-field errors) — drafts that the pre-revision lenient schema would have accepted. The smaller model timed out on one package in all repeats (including a 3600s re-attempt), a robustness observation reported as such. Wall-clock figures are CPU-bound and are not interpreted as productivity. De-identified human scoring against the published rubric (requirement fidelity, case relevance, editing effort) remains a manual gate; this experiment therefore supports the boundary claims — governance held on every run — and a honest negative-to-mixed drafting-quality signal, not a productivity claim. No controlled comparison with manual specification authoring, blinded expert-quality rating, token or energy cost analysis, or production error-rate estimate was performed. Accordingly, the LLM component is reported as an implemented, testable governance pattern, not as evidence that AI-generated specifications are faster or better.

4.5 RQ4: Exact replayability — capsules reproduce from seeds and manifests

Success criterion (predefined). Every public capsule re-executes from the submitted bundle under a pinned environment, and the recorded seeds reconstruct identical inputs and machine-readable results.

Result. For the eight-subject synthetic dataset, the pooled Kaplan–Meier survival probabilities at times 4, 5, 6, 8, 9, 11, 12, and 15 were respectively 0.875, 0.750, 0.625, 0.500, 0.375, 0.250, 0.250, and 0.250. The single-covariate Cox fit returned coefficient 0.0032679947, standard error 0.8284160536, and partial log-likelihood -9.9114479412 , all within the prespecified 10^{-10} absolute and relative tolerance. All 50 seeded row permutations reproduced the same quantities within

tolerance. Seed fidelity meets the criterion across the capsules: the jsonlite 50-case relation replays from its recorded seed with the first violation at case 1 unchanged, the survival known answers were additionally verified under both survival 3.8-3 and a later local version, and the same canary specification reproduces the pristine 11/11 result and each variant’s targeted failure. S1 supplies the dataset, independently derived expected and observed values, package versions, and replay code; SHA-256 manifests bind the replayed bytes. The example demonstrates exact traceability for one common clinical-analysis pattern; it does not validate the survival package generally.

4.6 RQ5: Mutation-funnel operability — predefined accounting

Success criterion (predefined). On a declared intended-use source scope with the green pre-condition satisfied, the funnel executes end to end: candidate count, seed, selected locations, and per-mutant verdicts are retained; invalid/stillborn and timeout outcomes are recorded as such and never credited as detections; survivors remain equivalence-undetermined; and the descriptive ratio is computed only over decided, executable mutants.

Protocol. The corrected mutation path is treated separately (Figure 3). Enabled specs must state package-relative source files and a scope rationale; outputs retain candidate count, seed, selected locations, and per-mutant verdicts. The six invalid/stillborn classes (Section 3.5) are never credited as detections, and the ratio remains conditional on the declared scope, operators, seed, and executable mutants.

Result. A dedicated public funnel run (supplement S5) executed the full flow on the synthetic canary package and on one real statistical package, each with a green pristine suite. On the canary (five candidates, full pool): detected 3, survived 1, invalid/stillborn 1, timeout 0, conditional ratio 0.75; the single survivor was reviewed as a weak-test case (an untested default-argument path), not an equivalent mutant. On the real package with a declared serialization scope (96 candidates, 20 sampled at a fixed seed): detected 1, survived 16, invalid/stillborn 3, timeout 0, conditional ratio 0.059 over 17 executable mutants; survivor review attributed 10 to weak tests (untested branches and boundary conditions the intended-use spec does not cover) and 6 to semantically equivalent or unobservable edits. Runs used the identical engine build (file hashes published in S5) on two platforms (Windows and a Linux container; package compilation constraints differed), so counts are platform-conditional exactly as declared. The portable S1/S3 replays disable source-package mutation because repeated source installation is platform-dependent; classifier behavior is supported there by unit and contract tests rather than an effectiveness fraction. Automatic equivalent-mutant classification and a platform-independent source-install replay remain unavailable, so the reviewed survivors stay recorded as equivalence-determined only by human inspection, and no general detection probability is claimed.

5 Threats to Validity

- **Mutation uncertainty.** Four syntactic operators do not represent real fault distributions. Equivalent mutants can lower a ratio and misclassified crashes raise it. Invalid/stillborn and timeout outcomes are separated, but call-graph proof and automatic equivalence detection are absent; no effectiveness fraction is reported.
- **Suggests dependencies are deliberately not installed** in the sandbox; check and test paths that require them degrade or are skipped. Residual risk: behavior reachable only through suggested packages is under-validated.

- **Spec quality is human-bound.** Schema validity and evidence binding cannot establish that requirements are sufficient; a thin spec can pass. Competent independent review remains necessary.
- **Characterization capture cannot prove correctness.** A golden-master value captured from the package under test only freezes current behavior; agreement with such a baseline is circular as a correctness argument and cannot expose a fault already present when the value was captured. The mandatory `oracle_source` declaration makes this provenance explicit: the public survival capsule uses `independent_derivation` for its Kaplan–Meier and Cox known answers (GM-001/GM-002), the jsonlite capsule uses `characterization_capture` for its implementation-specific serialization master (GM-001), and `second_implementation` is implemented and schema-tested but not exercised in the public capsules. Human sign-off of every captured baseline is the compensating control.
- **LLM drafting efficacy is unevaluated.** Contract tests establish that the drafting component cannot directly populate oracles, approve specs, or issue verdicts, but they do not establish semantic quality. Models can omit requirements or propose unsafe calls, and version changes can alter drafts. Retained provenance supports reconstruction, not correctness. Expert-controlled comparison is future work.
- **One small statistical example.** The survival capsule uses eight synthetic subjects and one Cox model; it does not establish performance across estimators, stochastic algorithms, or packages.
- **Single-author development and adjudication.** One person authored the specifications, built the prototype, executed the replays, and adjudicated the public results; the public capsule contains no independent second review of the kind the method itself prescribes. The evidence path is therefore self-reviewed, and the human-governance loop is demonstrated as a mechanism, not as an organizational practice.
- **The evaluation is a research prototype, not a regulated qualification.** Public local replays establish the reported numerical and injected-defect outcomes. They do not exercise enterprise identity, a qualified production environment, or legally attributable electronic signatures, and no such claim is made.
- **Egress restriction is application-layer.** Assessed-package code could open arbitrary outbound connections at run time; the declared mitigation (egress proxy) is designed, not implemented.
- **Package management is out of scope.** Dependency resolution and snapshot availability remain delegated to a pinned upstream host.
- **Counterexample replay depends on generator determinism.** Seeded generators make cases replayable, but a generator change across engine versions could shift input distributions; generator versions are part of the environment fingerprint.

6 Related Work

Software-test oracles and generated tests. The oracle problem and its solution families are reviewed by Barr et al.⁸ Metamorphic testing was introduced by Chen et al.,⁹ and later empirical work and surveys examined its effectiveness and application breadth.^{10, 11} It has also been applied directly to statistical software, including uniformity statistics.²⁶ The present prototype packages a

small, fixed relation library with seeded replay and review records; it does not establish that the four relations cover all relevant statistical transformations. Property-based testing follows the QuickCheck tradition¹⁸ and is available in the R ecosystem through libraries such as {hedgehog};¹⁹ the prototype instead executes deterministic seeded randomized property checks with its own fixed-enumeration generators. The invalid-input cases are closer to a small deterministic fuzz corpus than to modern coverage-guided fuzzing.²²

Generative assistance under deterministic gates. LLM-based code and test generation introduces a new oracle and provenance problem: a syntactically plausible expected value is not independent evidence.²⁰ The pattern evaluated here deliberately assigns the model no authority over expected values or disposition. Instead, package execution, schema checks, and qualified human approval remain deterministic or accountable gates, while prompts and drafts are retained as versioned evidence. The present study evaluates enforcement of that separation, not comparative model quality. Governance and risk frameworks are emerging in parallel: the NIST Generative AI Profile enumerates risk categories and controls for generative AI,¹⁶ and FDA has issued draft guidance on the use of AI to support regulatory decision-making for drug and biological products¹⁷ — both are cited here as risk-framing context, and no conformance to either is claimed. In the pharmaceutical-statistics setting, peer journals now provide biostatisticians with tutorials on LLM use and its limits,²¹ while software-engineering research surveys LLM support across the development life cycle.²⁰ The non-authoritative-draft governance mode evaluated here — structure generation permitted, evidence, approval, and verdicts withheld from the model — aligns with the caution these sources recommend.

Mutation analysis. Mutation testing is a mature test-adequacy technique,⁶ but equivalent-mutant identification remains a central interpretive problem.⁷ The implementation here adds a green precondition and an auditable survivor list. Its zero-dependency design is operationally useful, but the study does not compare its operators or results with established mutation tools, real faults, or manually classified equivalent mutants.

R package validation practice. The R Foundation documents R’s development, testing, and release controls,² while the R Validation Hub white paper and tools ({riskmetric}, {riskassessment}, and {valtools}) support package-level risk assessment and documentation.^{3, 4, 27, 28} Wider software-engineering work has also argued for stronger, trustworthy R package practices.²⁹ PHUSE guidance covers containerization, lock files, and snapshot repositories,⁵ and recent pharmaceutical literature frames reliability and sustainability as prerequisites for wider open-source adoption.¹ The prototype complements these approaches with intended-use cases and mutation diagnostics; it does not replace package risk assessment, system validation, or qualified human release decisions.

7 Conclusion

Intended-use package evidence should connect versioned requirements to executed tests and should also examine whether those tests respond to selected code changes. This study shows a feasible prototype using complementary test techniques, isolated execution, human review, and hash-linked records. It also provides an optional local LLM path intended to reduce the blank-page burden of specification drafting without allowing the model to create expected values, approve evidence, or issue a verdict; the authority boundary is tested, while productivity and draft quality remain unevaluated. Every captured golden-master value is a declared characterization baseline: non-circular evidence comes from independently derived known answers, oracle-free metamorphic relations, and human sign-off, not from the capture itself. The public canary shows that four prespecified source defects trigger their targeted requirements, while the jsonlite example shows that a documented

package default can be a calling- contract mismatch rather than a software defect. A synthetic Kaplan–Meier and Cox-model capsule demonstrates how the method applies to a pharmaceutical analysis and is independently replayable. Invalid-mutant handling and source scope have been corrected; the dedicated mutation-funnel run (RQ5) is pending, and no mutation effectiveness or regulatory compliance claim is made.

Acknowledgments and AI-Assistance Disclosure

AI assistance (a large language model coding assistant) was used in drafting this paper and developing prototype components, under author review. Separately, the evaluated software includes the optional local LLM specification-drafting component described in Section 3.2; the deterministic validation engine itself does not use AI. Empirical claims are supported by the local public replay artifacts. Manuscript assistance is not evidence for the evaluated LLM component, and the two uses are kept distinct.

Data Availability Statement

All reported data are synthetic or generated from public CRAN packages. S1 provides replay code, inputs, outputs, and specifications for the jsonlite and survival findings; S2 documents the LLM authority boundary and contract-test scope; S3 provides the complete synthetic canary source, four injected variants, specification, and raw results; and S6 replays the audit-chain integrity checks with a six-class tamper test set. A SHA-256 manifest binds the public capsule. Deployment, identity integration, and the full implementation are not required to reproduce the reported numerical claims and are withheld while the author retains commercial-development options. No credentials, patient data, or employer records are used.

Conflict of Interest

This work was conducted as independent personal research without external funding. The author retains rights in the prototype and may pursue future commercial development; this potential interest is disclosed. No vendor, employer, or patient data were used in the reported evaluation.

References

- [1] Sabanés Bové D, Seibold H, Boulesteix A-L, et al. The statistical software revolution in pharmaceutical development: challenges and opportunities in open source. *Drug Discov Today*. 2026;31(2):104613. doi:10.1016/j.drudis.2026.104613
- [2] R Foundation for Statistical Computing. *R: Regulatory Compliance and Validation Issues: A Guidance Document for the Use of R in Regulated Clinical Trial Environments*. March 12, 2026. <https://www.r-project.org/doc/R-FDA.pdf>
- [3] Nicholls A, Bargo PR, Sims J; R Validation Hub. *A Risk-Based Approach for Assessing R Package Accuracy Within a Validated Infrastructure*. January 23, 2020. Accessed September 5, 2026. <https://pharmar.org/white-paper/>
- [4] R Validation Hub, Kelkhoff D, Gotti M, et al. *riskmetric: Risk Metrics to Evaluating R Packages*. Version 0.2.7. CRAN; 2026. doi:10.32614/CRAN.package.riskmetric

- [5] PHUSE Data Visualisation & Open Source Technology Working Group. *Open Source Technology in Clinical Data Analysis*. Draft edition. Accessed September 5, 2026. <https://phuse-org.github.io/OSTCDA/>
- [6] Jia Y, Harman M. An analysis and survey of the development of mutation testing. *IEEE Trans Softw Eng.* 2011;37(5):649–678. doi:10.1109/TSE.2010.62
- [7] Madeyski L, Orzeszyna W, Torkar R, Józala M. Overcoming the equivalent mutant problem: a systematic literature review and a comparative experiment of second order mutation. *IEEE Trans Softw Eng.* 2014;40(1):23–42. doi:10.1109/TSE.2013.44
- [8] Barr ET, Harman M, McMin P, Shahbaz M, Yoo S. The oracle problem in software testing: a survey. *IEEE Trans Softw Eng.* 2015;41(5):507–525. doi:10.1109/TSE.2014.2372785
- [9] Chen TY, Cheung SC, Yiu SM. *Metamorphic Testing: A New Approach for Generating Next Test Cases*. HKUST Technical Report HKUST-CS98-01; 1998.
- [10] Liu H, Kuo F-C, Towey D, Chen TY. How effectively does metamorphic testing alleviate the oracle problem? *IEEE Trans Softw Eng.* 2014;40(1):4–22. doi:10.1109/TSE.2013.46
- [11] Chen TY, Kuo F-C, Liu H, et al. Metamorphic testing: a review of challenges and opportunities. *ACM Comput Surv.* 2018;51(1):4:1–4:27. doi:10.1145/3143561
- [12] Electronic Records; Electronic Signatures, 21 CFR part 11 (2026).
- [13] U.S. Food and Drug Administration. *Part 11, Electronic Records; Electronic Signatures—Scope and Application*. August 2003. Accessed September 5, 2026. <https://www.fda.gov/regulatory-information/search-fda-guidance-documents/part-11-electronic-records-electronic-signatures-scope-and-application>
- [14] International Society for Pharmaceutical Engineering. *GAMP 5: A Risk-Based Approach to Compliant GxP Computerized Systems*. 2nd ed. ISPE; 2022.
- [15] U.S. Food and Drug Administration. *General Principles of Software Validation: Final Guidance for Industry and FDA Staff*. January 2002. Accessed September 5, 2026. <https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-principles-software-validation>
- [16] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0): Generative AI Profile*. NIST AI 600-1; 2023. doi:10.6028/NIST.AI.600-1
- [17] U.S. Food and Drug Administration. *Considerations for the Use of Artificial Intelligence to Support Regulatory Decision-Making for Drug and Biological Products*. Draft guidance; 2025. Accessed September 5, 2026. <https://www.fda.gov/regulatory-information/search-fda-guidance-documents/considerations-use-artificial-intelligence-support-regulatory-decision-making-drug-and-biological-products>
- [18] Claessen K, Hughes J. QuickCheck: a lightweight tool for random testing of Haskell programs. In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*. ACM; 2000:268–279. doi:10.1145/351240.351266

- [19] Campbell H. *hedgehog: Property-Based Testing*. Version 0.2. CRAN; 2025.
doi:10.32614/CRAN.package.hedgehog
- [20] Hou X, Zhao Y, Liu Y, et al. Large language models for software engineering: a systematic literature review. *ACM Trans Softw Eng Methodol*. 2024;33(8):1–79. doi:10.1145/3695988
- [21] Dobler D, Binder H, Boulesteix A-L, et al. ChatGPT as a tool for biostatisticians: a tutorial on applications, opportunities, and limitations. *Stat Med*. 2025;44(23–24). doi:10.1002/sim.70263
- [22] Manès VJM, Han HS, Han C, et al. The art, science, and engineering of fuzzing: a survey. *IEEE Trans Softw Eng*. 2021;47(11):2312–2331. doi:10.1109/TSE.2019.2946563
- [23] Haber S, Stornetta WS. How to time-stamp a digital document. *J Cryptol*. 1991;3(2):99–111. doi:10.1007/BF00196791
- [24] Therneau T. *A Package for Survival Analysis in R*. Version 3.8-3. CRAN; 2025.
<https://CRAN.R-project.org/package=survival>
- [25] Ooms J. *jsonlite: A Simple and Robust JSON Parser and Generator for R*. Version 2.0.0. CRAN; 2025. doi:10.32614/CRAN.package.jsonlite
- [26] Patel K, Hierons RM. A partial oracle for uniformity statistics. *Softw Qual J*. 2019;27(4):1419–1447. doi:10.1007/s11219-019-09459-0
- [27] Clark A, et al. *riskassessment: Risk Assessor for R Packages*. CRAN. Accessed September 5, 2026. doi:10.32614/CRAN.package.riskassessment
<https://pharmar.github.io/riskassessment/>
- [28] Hughes E, Miller E, Vendettuoli M, Eshghi P. *valtools: Automate Validated Package Creation*. CRAN. Accessed September 5, 2026. doi:10.32614/CRAN.package.valtools
<https://pharmar.github.io/valtools/>
- [29] Landau WM. We need trustworthy R packages. *R J*. 2021;13(2):15–17. doi:10.32614/RJ-2021-109