Review Methodology for AI-Assisted Blog Post Drafting

Ronald G. Thomas

2026-08-18

A post outlined by a human and drafted by an assistant still needs one owner who can defend every paragraph: this is the checklist for becoming that owner.

Introduction

A common drafting pattern now pairs a human author, who supplies the topic, argument, and intended audience, with an AI assistant that produces the prose itself, section by section from an outline or a set of talking points. The arrangement speeds up drafting considerably, but it creates the same responsibility gap that AI- assisted code development creates. The byline carries the author's name, and with it the obligation to stand behind every claim, citation, and turn of phrase, regardless of who typed it first.

This post presents a methodological framework for closing that gap in the specific case of blog prose: a structured review process for posts drafted through human-AI collaboration. It is a direct transposition of a companion framework originally written for reviewing AI-assisted R packages, Code Review for AI-Assisted R Package Development, onto prose rather than code. Where that post asks whether a function does what it claims, this one asks whether a paragraph does what it

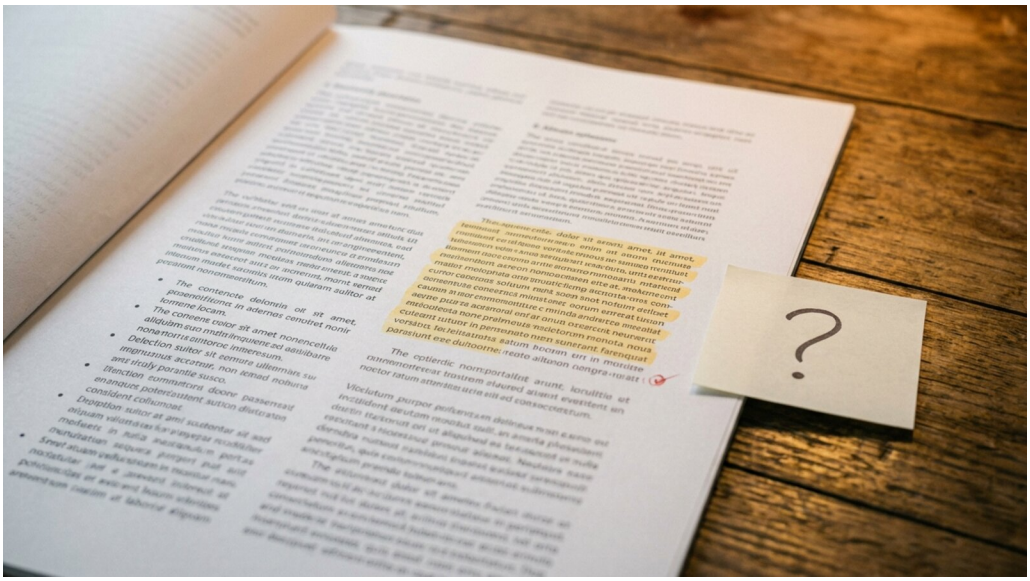
claims. Both end the same way: with an explicit ownership test rather than a vague sense that the piece "reads fine."

Motivations

- **The responsibility gap is real for prose, not just code.** An author who cannot defend a factual claim in their own post cannot answer a reader's correction, retract cleanly, or extend the argument in a follow-up.
- **AI-drafted prose has characteristic failure modes.** Hallucinated citations, generic voice, and structural padding do not show up the same way in a human-drafted first pass, so treating the draft like ordinary self-edited copy misses them.
- **A polished sentence is not a checked sentence.** A paragraph can read smoothly and still assert something false, unsourced, or inconsistent with an earlier section.
- **Ownership needs a definition, not a feeling.** "I read it over" and "I can restate every claim in this post from memory, and defend each one" are different claims, and only the second one licenses publishing under your own name.

Objectives

1. Define the five review phases (structural, paragraph-by-paragraph, verification, narrative integration, publication safety) and what each phase is for.
2. Catalog the AI-generated prose patterns that deserve specific scrutiny during the paragraph-level pass.
3. Provide a concrete note-taking and severity-classification system for tracking findings through revision.
4. Establish a verifiable test for "ownership": what an author must be able to do, unaided, before publishing the post as their own.



What Is This Framework For?

The scope is deliberately narrow. It addresses review of blog posts where a human author defined the topic, argument, structure, and intended audience, an AI assistant drafted the prose from that outline, and the author now seeks complete ownership and understanding of the result before it publishes. It is not a general copyediting guide, and it does not replace a style guide or a platform's editorial checklist; it sits alongside both.

A thorough review in this context serves five purposes:

1. **Comprehension.** The author understands and can restate every claim.
2. **Correctness.** Every factual assertion holds up.
3. **Voice.** The prose reads as the author's own, not as generic assistant output.
4. **Safety.** The piece introduces no legal, ethical, or reputational exposure.
5. **Alignment.** The published argument matches the outline the author actually intended, rather than drifting toward whatever the assistant found convenient to write.

Prerequisites

Before starting, assemble the original outline or talking points, any source material the post draws on (papers, documentation, personal notes, prior posts), the conversation log or prompts used during drafting, and the site's own style guide or `CLAUDE.md` conventions. Then set up a working environment: a rendered preview of the post (`quarto render index.qmd`), a spell-checker and prose linter if the toolchain has one, and a separate browser tab for verifying every external link and citation as it comes up, rather than batching link-checking to the end.

The Five Review Phases

Phase 1: Structural Review

Begin at the post level, before reading any individual paragraph.

Front matter. Confirm the YAML is complete and correct: title, date, categories, description, image, and any site-specific fields. A description that oversells the post, or categories that do not match the actual content, are structural defects even before the prose is read.

Outline fidelity. Compare the drafted section headings against the original outline. Sections that were not requested, sections that were requested but are missing, and sections reordered without reason are all worth flagging before line-level review starts.

Length and pacing. Is the post proportionate to its topic? A short claim padded to meet an assumed word count, or a substantial claim compressed to a single paragraph, both signal that the assistant optimized for a shape rather than for the content.

Phase 2: Paragraph-by-Paragraph Review

This is the core of the process. For every paragraph, work through five questions in order:

Correctness. Is every factual claim accurate, and is every claim that needs a source actually sourced? Does the paragraph overstate its confidence relative to the evidence behind it?

Argument alignment. Does the paragraph advance the argument the author intended, or has the assistant introduced a tangent, a hedge, or a conclusion that was not requested?

Attribution. Are quotations, statistics, and specific claims attributed to a real, checkable source? Is anything presented as common knowledge that is actually a specific claim needing a citation?

Voice and register. Does the paragraph sound like the author, in the author's usual register, or does it read as generic explanatory prose that could have been written about any topic?

Clarity. Can you explain, in your own words, what the paragraph is claiming and why it belongs in this position in the post?

For each section, read the outline note before reading the drafted prose, trace the argument as a reader unfamiliar with the topic would encounter it, and verify that every specific claim has a source you have personally checked. Deliberately question any transition that resolves too smoothly: AI-drafted prose tends to paper over gaps in the argument with a fluent connective phrase.

Patterns specific to AI-generated prose

A handful of patterns recur often enough in assistant-drafted posts to warrant a dedicated pass:

- **Hallucinated citations.** A source, statistic, or quotation that does not exist, or that exists but does not say what the paragraph claims it says.
- **Generic voice.** Prose that is fluent and correct but indistinguishable from a thousand other posts on the same topic, carrying none of the author's actual perspective.
- **Structural padding.** Bullet lists, subheadings, or transitional sentences added to meet an assumed shape rather than because the content needed them.
- **Overqualified hedging.** Reflexive qualifiers ("it is worth noting," "arguably," "in many cases") that dilute a claim the author actually holds with confidence.
- **Confident overreach.** The mirror failure: a claim stated with more certainty than the underlying evidence supports.
- **Repetition across sections.** The same point restated in slightly different words in two places, because each section was drafted somewhat independently of the others.

Phase 3: Verification Pass

Coverage of "the post has citations" is a starting point, not a conclusion. For every factual claim, statistic, and quotation, separately verify the source exists, says what the post claims it says, and is the most current or authoritative version available. For every external link, confirm it resolves and points to the intended target rather than a redirect or an archived page. For every code example, run it against the version of the software the post claims to describe, since a code block that merely looks plausible is the prose equivalent of a test that asserts nothing.

Then check correspondence in both directions: every claim in the post should trace to a source you have checked, and every source gathered during research should be reflected accurately somewhere in the post. A source cited to support a stronger claim than it actually makes is as much a defect as a claim with no source at all.

Phase 4: Narrative Integration Review

Trace the argument across section boundaries rather than within a single section. Does each section build on the one before it, or could sections be reordered without loss? Identify any claim made early in the post that a later section quietly contradicts or narrows. Verify that the opening promises what the post actually delivers, and that the closing genuinely follows from the body rather than introducing a new claim for the first time in the conclusion.

Phase 5: Publication Safety Review

Examine the post for the standard exposure classes specific to published writing:

- Claims about identifiable people or organizations that could be defamatory if false.
- Personally identifiable information used without consent, even in an anonymized-seeming example.
- Quoted material used beyond fair use without permission or attribution.
- Technical claims about a product or vulnerability that require responsible disclosure rather than public posting. Finally, consider licensing: are any images, code snippets, or quoted passages used under a license the post's format allows, and is that license recorded where readers can find it?

Documentation Review

Two audiences get separate passes. For the reader arriving cold, the opening paragraphs should state the topic and the post's claim clearly, without requiring the rest of the post to disambiguate what the piece is actually about. For a reader returning to cite the post later, headings should be specific enough to serve as anchors, and any code or configuration shown should be reproducible from the post alone, without requiring context from a conversation the reader was not part of.

Annotation and Severity

Maintain structured notes during the review rather than relying on memory. A minimal per-section template:

```
## Section: section_heading

### Status: [Reviewed | Needs Revision | Approved]

### Understanding
[Summary of the section's claim, in your own words]

### Concerns
- [Issue 1]
- [Issue 2]

### Questions
- [Question for further investigation]

### Changes Required
- [ ] Change 1
- [ ] Change 2
```

Classify every finding by severity: **Critical** (a false factual claim, a fabricated source, or a defamation or privacy risk), **Major** (a claim that overstates its evidence, a significant departure from the intended argument, or a broken code example), **Minor** (an awkward transition, a voice inconsistency, or a formatting slip), or **Enhancement** (a suggestion beyond what the outline called for). The severity tier determines the order of revision, not whether an issue gets fixed at all.

Remediation and Final Verification

Work through issues in severity order: critical issues affecting factual accuracy or legal exposure first, then major issues affecting the argument's soundness, then minor issues and enhancements as time allows. For each fix, understand why the original claim was wrong or misaligned before rewriting it, rewrite the passage yourself rather than asking the assistant to patch it without understanding

the patch, re-verify any source the fix touches, and re-read the surrounding paragraphs to confirm the fix did not break a transition.

After remediation, re-read the entire post start to finish as a reader would, confirm every link still resolves, and review the diff against the last reviewed draft before considering the review closed. Before calling the review complete, run the full verification pass: reread every citation against its source one final time, re-run every code example, and confirm the rendered preview matches the source `.qmd` with no formatting artifacts.

Per-Post Review Checklist

The five phases above are the reasoning behind this checklist; the checklist itself is what to actually run through for a given post. Copy it per post rather than trying to hold it in memory across multiple drafts in progress at once.

Phase 1: Structural

- ☐ YAML front matter complete and accurate (title, date, categories, description, image)
- ☐ Section headings match the original outline, with no unrequested additions or unexplained omissions
- ☐ Length and pacing proportionate to each claim, not padded or compressed to fit an assumed shape

Phase 2: Paragraph-by-paragraph

- ☐ Every paragraph reviewed for correctness, argument alignment, attribution, voice, and clarity
- ☐ Every specific claim traced to a source you have personally checked
- ☐ Every AI-generated-prose pattern in this post's list (hallucinated citations, generic voice, structural padding, overqualified hedging, confident overreach, repetition across sections) checked for, not just read past

Phase 3: Verification

- ☐ Every factual claim, statistic, and quotation verified against its actual source, not just checked for the presence of a citation
- ☐ Every external link confirmed to resolve to the intended target
- ☐ Every code example run against the software version the post claims to describe

Phase 4: Narrative integration

- ☐ Each section confirmed to build on the one before it, not independently reorderable without loss
- ☐ No early claim quietly contradicted or narrowed by a later section
- ☐ Opening promise and closing conclusion both checked against what the body actually delivers

Phase 5: Publication safety

- ☐ No claim about an identifiable person or organization that could be defamatory if false
- ☐ No personally identifiable information used without consent, even in an anonymized-seeming example
- ☐ Quoted material, images, and code snippets confirmed to be within license or fair use, with the license recorded

Before publishing: ownership sign-off

- ☐ Can restate the post's central argument and every section's claim without referring to the draft
- ☐ Can name the source for every fact or statistic in the piece
- ☐ Can identify any sentence that does not sound like your own voice
- ☐ Can answer a skeptical reader's challenge to any specific claim without looking it up again
- ☐ Can respond to a comment or correction and defend the piece if a cited source is later challenged

A "no" anywhere in this checklist is a blocker, not a note for later; see Establishing Ownership below for what each unchecked box actually costs if it ships unresolved.

Things to Watch Out For

- **Fluency as a proxy for accuracy.** A sentence that reads smoothly has passed no factual check at all.
- **Delegating the fix back to the assistant.** Asking the assistant to correct a flagged claim and then accepting the correction unread repeats the original problem one level down.
- **Treating a hallucinated citation as a rare edge case.** It surfaces often enough in assistant-drafted prose that a dedicated existence check for every source is worth the time.
- **Reviewing in draft order instead of argument order.** A post's argument sometimes does not flow in the order it was drafted; review the logical dependency chain, not just the section order.
- **Skipping the "why," not just the "what."** A paragraph can be factually correct and still misaligned with the argument the author actually intended to make.
- **Conflating "read it" with "understood it."** The annotation template's "Understanding" field is not decorative; if you cannot fill it in from memory, the section has not actually been reviewed.

Lessons Learned

Conceptual understanding:

- Ownership of a published post is a testable claim, not a feeling of familiarity with the topic.
- Fluency and accuracy are different axes; a review needs both.
- AI-drafted prose has a distinct failure signature (hallucinated sources, generic voice, structural padding) that ordinary self-editing does not catch.

Technical:

- Verifying a citation means opening the source and checking it says what the post claims, not confirming a link merely resolves.
- Reading a post aloud, or through a text-to-speech tool, surfaces voice inconsistencies a silent read-through misses.
- A rendered preview catches formatting and link defects that a raw `.qmd` read-through will not.

Gotchas:

- A confident tone can mask a claim with no source at all; check the sourcing independently of how the claim reads.
- A code example that renders without error in the post's preview can still be wrong for the reader's actual environment; run it, do not just render it.
- A paragraph's claim can be accurate in isolation while contradicting an earlier section; verify both directions.

Limitations

- The framework is scoped to blog-style prose; the phase structure generalizes, but the specific failure patterns cataloged here were observed in short-to-medium-length posts, not book-length manuscripts. See the companion post Review Methodology for AI-Assisted Textbook Drafting for the longer-form adaptation.
- It assumes the human author already has enough domain fluency to evaluate factual accuracy; it is not a substitute for domain expertise.
- It does not specify how much reviewer time is proportionate to a post's length or stakes; that judgment is left to the reviewer.
- The ownership test in the final section is self-administered; it provides no external verification that the answers given are accurate.

Opportunities for Improvement

1. A companion checklist scaled down for short, low-stakes posts, where the full five-phase process is disproportionate.
2. Worked severity-classification examples, since "Major" versus "Critical" boundary cases (an overstated claim versus a false one) are where reviewers disagree most.

3. Guidance on reviewing an AI-assisted *revision* to an already-owned post, as distinct from reviewing an entire AI-drafted post from scratch.
4. A shared vocabulary for disclosing AI involvement in a post's footer or metadata, referenced in this framework's publication safety phase but not specified in detail.
5. An adaptation of the framework for series posts, where narrative integration must be checked across posts, not only within one.

Establishing Ownership

Before publishing the post, verify that you can do the following without referring to the draft. Restate the post's central argument and every section's claim in your own words, and name the source for every fact or statistic in the piece. Identify which sentences do not sound like your own voice, and answer a skeptical reader's challenge to any specific claim without looking it up again. Separately, confirm publication readiness: can you respond to a comment or correction, issue a retraction or update if a claim turns out to be wrong, and defend the piece if a source you cited is later challenged?

Consider also documenting the drafting process itself: that AI assistance was used, what review process was undertaken, and, where the platform supports it, a note in the post's footer disclosing the extent of that assistance. This is not required by most publishing platforms, but it is honest, and it gives a future reader, or a future version of the author, the context this review process assumed from the start.

Review does not end at first publication. Any substantive AI-assisted edit to a published post should go through the same review before it goes live, and a post's factual claims are worth periodic re-verification as sources age or are superseded. Knowledge decays without use: revisiting the post's claims occasionally, and keeping review notes accessible, are what keep the ownership claim true over time, not just true at the moment of first publication.

Wrapping Up

What Did We Learn?

Thorough review of an AI-drafted blog post requires the same rigor a careful editor would bring to any submitted piece, plus a specific sensitivity to the failure modes assistant-drafted prose tends to produce. The process demands more than a read-through; it requires achieving genuine understanding of every claim before accepting responsibility for it under a byline. The five-phase structure (structural, paragraph-by-paragraph, verification, narrative integration, publication safety) gives that process a checklist rather than leaving it to instinct, and the ownership test at the end gives it a stopping condition: review is complete when the answers to the ownership questions are honestly yes, not when the reviewer is tired of reading prose.

Main takeaways:

- Fluency is a floor, not a target; verify sources, do not just read smoothly past them.
- AI-drafted prose has a specific set of failure patterns worth a dedicated review pass.

- Ownership is demonstrated by restatement and unaided defense, not claimed by completing a checklist.

If you are trying this yourself: start with the structural review before reading any paragraph closely, keep the annotation template open in a second window rather than trying to hold findings in memory, and do not flip a post to "reviewed" until you can honestly answer every question in the Establishing Ownership section.

See Also

This post is a direct transposition of a companion framework written for a different medium: Code Review for AI-Assisted R Package Development, which applies the same five-phase structure and ownership test to R package code rather than prose. Two further siblings extend the same framework to longer or more constrained forms: Review Methodology for AI-Assisted Textbook Drafting for book-length, pedagogically structured manuscripts, and Review Methodology for AI-Assisted Research Paper Drafting for academic manuscripts, which adds a compliance dimension grounded in published journal and publisher AI-disclosure policy.

Key resources:

- The Elements of Style — Strunk and White
- On Writing Well — William Zinsser

Reproducibility

Source document: adapted from `analysis/report/index.qmd` in the sibling post `rp-code-review-methodology` (`~/prj/rgtlab/posts/rp-code-review-methodology/`). This post transposes that framework's five-phase structure and ownership test from code review onto prose review; no methodological claims specific to R packages were carried over.

Session information:

```
R version 4.6.1 (2026-06-24)
Platform: aarch64-apple-darwin25.4.0
Running under: macOS Tahoe 26.6.2

Matrix products: default
BLAS:   /opt/homebrew/Cellar/openblas/0.3.34/lib/libopenblasp-r0.3.34.dylib
LAPACK: /opt/homebrew/Cellar/r/4.6.1/lib/R/lib/libRlapack.dylib;  LAPACK version 3.12.1

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Los_Angeles
```

```
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.6.1  fastmap_1.2.0   cli_3.6.6       tools_4.6.1
 [5] htmltools_0.5.9 parallel_4.6.1  otel_0.2.0      yaml_2.3.12
 [9] rmarkdown_2.31  knitr_1.51      jsonlite_2.0.0  xfun_0.58
[13] digest_0.6.39   rlang_1.3.0     evaluate_1.0.5
```

Rendered on 2026-08-18 at 12:01 PDT. Source: ~/prj/rgtlab/posts/rp-blog-review-methodology/analysis/report/ind

---

Let's Connect

Questions, corrections, or a different view on where this framework is too strict or too loose are welcome in the comment thread below.

- **GitHub**: rgt47
- **Email**: Contact form