Paper ML24

Using ChatGPT for AI-assisted Quality Control of Clinical Trial Tables and Figures: From Proof of Concept to an Integrated Workflow

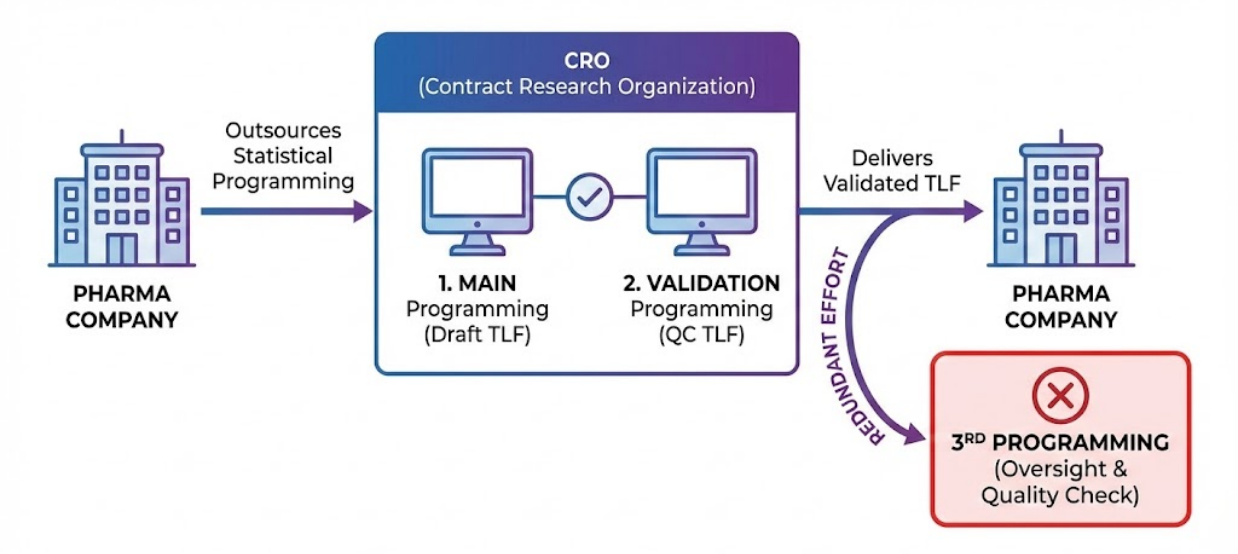
Unnat Patel, Brad Moore, Jenny Price, Brent Burger, UroGen Pharma
Kate Roth, Louisiana State University

ABSTRACT

Many sponsors outsource statistical programming yet remain responsible for the quality of tables, listings, and figures (TLFs). This session presents a no-code QC approach that uses ChatGPT Enterprise to recreate TLFs from Analysis Data Model (ADaM) datasets and compare them with vendor outputs across all TLFs. Prompts are based on shells and analysis notes; ChatGPT Enterprise generates independent reproductions and highlights differences for review. Each run records prompt text, data version, and model version to maintain traceability. The focus is practical: how to structure prompts, prepare inputs, log runs, and review differences — without writing code. Attendees take away reusable prompt templates, a simple run log format, and a reviewer checklist they can apply.

INTRODUCTION

Outsourcing Biometrics functions has become routine for many sponsors. Contract research organizations (CROs) provide specialized expertise and scalable delivery models, but sponsors remain accountable for the quality and integrity of deliverables (SDTM/ADaM datasets and TLFs). Oversight is typically implemented through document/code review, log review, and selective independent double-programming of critical endpoints; however, these approaches can be resource intensive and difficult to apply consistently across large portfolios or compressed timelines.

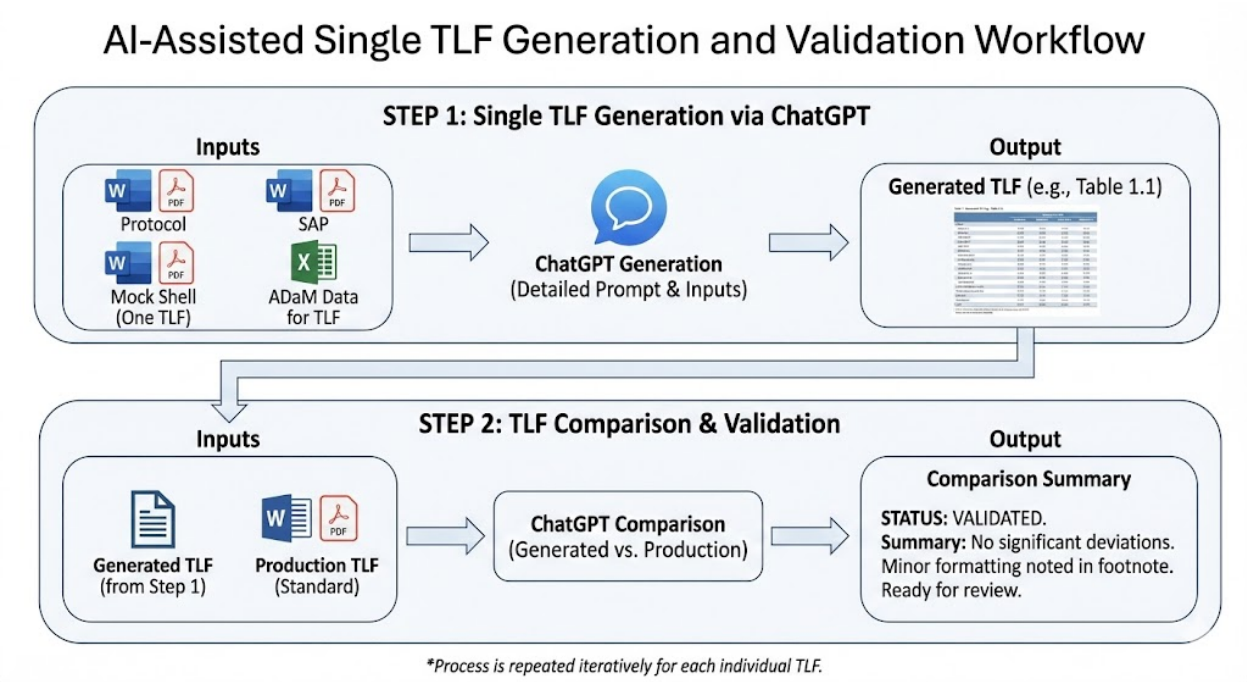


Generative AI tools such as ChatGPT Enterprise (*Wikipedia contributors, 2025*) offer a potential new modality for programming QC. Rather than writing explicit validation code, a sponsor can provide analysis datasets, specifications, and structured prompts to an AI system that independently regenerates selected TLFs and highlights

differences for review. Newer models are described as handling complex, multi-step tasks more effectively (*Welch, 2025*), which is attractive for ADaM-driven "no-code" QC scenarios that require interpreting dataset structures and written specifications.

These capabilities prompted a central question: **Can ChatGPT act as an independent validator of TLFs by regenerating outputs directly from ADaM datasets and comparing them with vendor outputs?** We explored this through a proof of concept (PoC) exercise conducted using ChatGPT Enterprise to recreate selected TLFs from ADaM datasets and compare them with vendor outputs. Building on the PoC, we evolved the work into a governed, reusable workflow suitable for controlled use. This paper summarizes the workflow, shares key PoC and pilot results, and shows how AI-assisted QC can be embedded into existing, risk-based validation processes.

METHODS



Design principles

We defined the ChatGPT QC workflow as an ADaM-only, no-code pilot running in a controlled ChatGPT Enterprise workspace. The aim was not only to see whether TLFs could be reproduced, but also to define how an AI-assisted process could be embedded in a controlled environment before extending upstream.

Three design principles guided the approach:

- **ADaM-only starting point** – Only the ADaM datasets were loaded into ChatGPT as study data (with the Protocol, SAP, and mock shells provided as supporting reference documents), aligning with the "one PROC away" concept and keeping scope focused on the analysis layer. When TLFs could not be reproduced from ADaM alone, we treated this as a potential specification gap requiring attention.
- **Independent re-derivation** – ChatGPT was treated as a second implementation (analogous to dual programming), regenerating TLFs from ADaM and written specifications (protocol, SAP, shells) without access to vendor code.
- **Human-in-the-loop oversight** – AI supported QC, while a programmer or statistical reviewer evaluated discrepancies, resolved ambiguous specifications, and approved final conclusions and documentation.

Stage I – Proof of Concept

Stage I was a focused proof of concept (PoC) using one completed study and a small set of TLFs: a demographics table and a Kaplan–Meier (KM) plot. ADaM datasets (e.g., ADSL, ADTTE) were loaded into a ChatGPT Enterprise workspace to assess whether the model could regenerate vendor outputs.

Key questions were whether ChatGPT could correctly interpret analysis populations from ADSL flags and produce TLFs whose counts, percentages, and KM curves were consistent with vendor outputs. The PoC used simple free-text prompts rather than the structured templates developed later. Initial observations showed that output quality was primarily driven by prompt specificity: demographics tables reproduced well with clear instructions, while KM plots often required iteration to align time-to-event conventions and formatting.

Stage II – Generic Prompt Engineering and Validation

Based on Stage I findings, Stage II had two goals:

- Design generic prompt templates reusable across standard TLF types (e.g., demographics, AE/TEAEs, KM plots).
- Validate those templates using controlled ADaM extracts and approved production outputs.

Generic prompts were developed so the same core structure could be applied across TLFs while only swapping study-specific context. Templates define the model role, state the objective of reproducing one TLF at a time from study documentation and ADaM, constrain the data foundation to ADaM in Excel plus a PROC CONTENTS export (proc_contents.xlsx), and enforce a consistent stepwise workflow with self-checks and governance. The full template structure and example prompt snippets are provided in **APPENDIX 1**.

For sampling-based validation, a repeatable data flow was defined: ADaM extracts, mock shell, Protocol/SAP, and the appropriate generic prompt were loaded into ChatGPT; the model regenerated the tables or figures, and comparison files were produced for reviewer assessment. Formal test scripts were executed for a small set of outputs (demographics table using ADSL; AE/TEAE table using ADSL/ADAE; KM plot for duration of response using ADSL/ADTTE with the relevant PARAMCD). Test Scripts were executed multiple times under identical conditions, with environment details recorded, and each ChatGPT-generated output was compared against an approved production reference.

Observed differences were limited to explainable issues such as rounding-policy mismatches, ambiguity in titles/footnotes or SAP text, and technical limitations (e.g., generating “Page X of Y”). Discrepancies and operational issues were logged with an exception ID, root cause, and resolution/workaround in an exception log, and the same validation pattern can be reused when templates or model versions change.

Stage III – Workflow Integration

Following validation of the prompt templates and data flow, Stage III focused on integrating the ChatGPT no-code QC approach into an end-to-end workflow and governance model. This included defining a repeatable process for production use, updating the Statistical Programming Work Instruction to recognize AI-assisted independent QC within the existing risk-based framework, and piloting the approach on TLFs prepared for a pre-specified interim reporting event (e.g., demographics, disposition, response-rate, adverse-event-by-time-period summaries, TEAE and exposure tables).

The pilot highlighted specification ambiguities that required refinement; clarifying SAP language and/or table shell programming notes resolved discrepancies by making intended rules explicit rather than implied.

All execution records (prompt text, data version, model version, outputs), regenerated outputs, comparison files, and discrepancy dispositions were archived alongside traditional QC artifacts in a controlled repository. An exception-log template captured issues, root causes, and resolutions (e.g., rounding conventions, file-format preferences, filename handling, data-normalization rules), ensuring lessons learned were fed back into prompt templates and operating instructions. Focused knowledge-sharing sessions supported adoption while keeping human-in-the-loop oversight central to QC decision-making.

RESULTS

Accuracy and Feasibility

Across Stage I to Stage III, ChatGPT produced results comparable to vendor deliverables for most tables and figures on the first pass when prompts clearly specified the analysis population, denominators, and reporting conventions. Remaining differences were infrequent and were resolved through prompt iteration or clarification of ambiguous specifications—most often where conventions were implicit or under-specified. Repeated runs with fixed inputs and independent users produced stable results within expected rounding and formatting tolerances, supporting the feasibility of a no-code, ADaM-only QC approach.

Issues Observed

Across the PoC and pilot, most issues were driven by specifications, data coding, or prompt/template conventions—not by the model itself. They fell into four categories:

1) **Specification/documentation gaps –** Rules were implicit in shells/SAP text (e.g., time windows or denominators). Updating shells with explicit programming notes resolved such issues.
2) **ADaM or upstream coding/derivation issues** – The model followed written rules but exposed inconsistent data representations (e.g., empty strings instead of an explicit "Missing" category), resolved via explicit prompt instruction. These were evaluated on a case-by-case basis on whether to have vendor address with data updates.
3) **Prompt/template defects** – Missing reporting conventions (rounding, zero suppression, sort order, output formatting). Once stated explicitly in reusable templates, these differences typically disappeared and were captured as standard conventions/exceptions.
4) **LLM numeric/logical slips** – Rare cases where outputs were inconsistent despite correct inputs; handled like a programming bug (documented, investigated, and rechecked against reference outputs before sign-off).

Overall, issues overwhelmingly fell into categories 1) through 3), supporting the feasibility of a no-code, ADaM-driven QC approach.

Efficiency Gains

Once prompt templates were standardized, QC effort shifted away from programming toward review and investigation. Many TLFs could be validated with relatively little hands-on effort (attach ADaM, run the template, review the comparison), especially for experienced reviewers. For complex outputs, time spent on interpretation and root-cause analysis had comparable time trade-off to writing and debugging independent QC code. Because the incremental effort per additional unique TLF was low, QC coverage expanded to include lower-priority tables that are not typically double-programmed.

Documentation Quality

LLM-assisted drafting improved QC documentation quality by making summaries more structured and repeatable. Reviewers used execution records (including prompts and model outputs) to draft concise QC summaries (scope, methods, discrepancies, conclusions), then edited and approved them as usual. Reports were more uniform across studies and required less time drafting standard text.

DISCUSSION

This workflow was executed in a controlled ChatGPT Enterprise workspace, leveraging OpenAI's enterprise privacy commitments.

Benefits of an ADaM-Only Approach

We intentionally started with an ADaM-only scope to demonstrate an end-to-end, no-code QC workflow at the analysis layer before extending upstream. This boundary enabled vendor-independent reproduction (without CRO code) and supported reusable prompt templates that broaden QC participation beyond programmers.

Human Oversight Remains Essential

Even with newer models, ChatGPT remains probabilistic and can misinterpret specifications or adopt unintended assumptions. Human reviewers remain responsible for:

- **Interpreting specifications** – resolving vague or conflicting SAP and shell text using clinical and statistical judgment.
- **Assessing materiality** – distinguishing minor differences (e.g., rounding/formatting) from discrepancies with potential clinical or regulatory impact.
- **Governance and trust** – deciding when to use AI-assisted QC, handling model updates, and documenting AI involvement in a way that satisfies oversight and audit requirements.

Limitations

Key limitations of the current approach include:

- **Scope of validation** – The workflow focuses on output-level QC (consistency between TLFs, ADaM, and specifications) and does not fully validate ADaM derivations; upstream checks are still required.
- **Context window and scale** – Very large datasets or complex multi-page TLFs can approach model context limits, requiring sampling, splitting, or summarization.
- **Model opacity and evolution** – Model behavior may change as new versions are deployed, underscoring the need to record model version.

Generalization to complex inferential analyses – Extending beyond descriptive TLFs (e.g., to Mixed Model for Repeated Measures or survival models) would require additional prompt instruction, code-execution integration, and validation.

Future Work

We plan to extend the approach along two main fronts:

- QC using SDTM/raw data for selected TLFs by re-deriving or cross-checking key ADaM variables from SDTM using explicit rules before applying the existing ADaM-based TLF QC workflow.
- AI-assisted TLF development using a maker/checker loop in which drafts are proposed and independently challenged to improve efficiency while preserving human oversight.

CONCLUSIONS

This work shows that ChatGPT can function as an independent “virtual programmer,” regenerating TLFs from ADaM and highlighting differences relative to vendor outputs. In our PoC and pilot, this regenerate-and-compare workflow reduced redundant double programming, expanded more QC coverage to lower-priority TLFs, and improved traceability through standardized templates and execution records. AI-assisted quality control can effectively complement traditional validation when supported by strong governance, transparent documentation, and sustained human-in-the-loop oversight.

REFERENCES

Wikipedia contributors. (2025, November 19). ChatGPT. In Wikipedia, The Free Encyclopedia. Retrieved December 10, 2025, from https://en.wikipedia.org/wiki/ChatGPTWikipedia

Welch, N. (2025, August 18). ChatGPT 5.0: What’s new, key features, and how to use it. Vidatec. https://vidatec.com/learning/chatgpt-5-0-whats-new-key-features-and-how-to-use-it/Vidatec

CONTACT INFORMATION

Contact authors at:

Author Name: Unnat Patel
Company: UroGen Pharma
Address: 400 Alexander Park Dr. Princeton NJ, 08540
Email: unnat.patel@urogen.com

Author Name: Brad Moore
Company: UroGen Pharma
Address: 400 Alexander Park Dr. Princeton NJ, 08540
Email: brad.moore@urogen.com

Author Name: Jenny Price
Company: UroGen Pharma
Address: 400 Alexander Park Dr. Princeton NJ, 08540
Email: jenny.price@urogen.com

Author Name: Kate Roth
University: Louisiana State University
Email: kroth2424@gmail.com

Author Name: Brent Burger
Company: UroGen Pharma
Address: 400 Alexander Park Dr. Princeton NJ, 08540
Email: brent.burger@urogen.com

APPENDIX 1 – Prompt Template Structure and Example Snippets for ChatGPT TLF Generation

Prompt Template Structure:

Role

In this framework, the assistant plays the role of an expert biostatistical programmer in uro-oncology, with practical familiarity with FDA and PMDA expectations, CDISC SDTM and ADaM standards, TransCelerate guidance, and ICH E3/E6/E9. The goal of this role description is to make clear that every decision—population flags, endpoint handling, imputation, layout, and traceability—is grounded in established regulatory and industry conventions rather than ad hoc choices.

Prompt snippet (Role)

Act like an expert biostatistical programmer (uro-oncology) with deep knowledge of FDA/PMDA, CDISC SDTM/ADaM, TransCelerate, and ICH E3/E6/E9.

Objective

The objective is to replicate clinical tables, listings, and figures one at a time with high fidelity, using only study documentation (Protocol, SAP, TLF mock shells, annotated CRF) and ADaM datasets supplied in Excel together with a PROC CONTENTS export (proc_contents.xlsx). The engine is deliberately constrained to use ADaM as its sole data source, avoiding direct reliance on raw EDC or SDTM, so that it behaves like an analysis-layer consumer rather than a data-management system, while still producing outputs suitable for regulatory review.

Prompt snippet (Objective)

Objective: Replicate one clinical TLF at a time with high fidelity using only session uploads (Protocol, SAP, Mock Shell, proc_contents.xlsx). Focus on ADaM data in Excel (PROC CONTENTS + data) and do not use raw EDC or SDTM.

Data Foundation: ADaM in Excel

All computations are driven from ADaM-level datasets and their metadata, both delivered in Excel form. The PROC CONTENTS export (proc_contents.xlsx) provides one row per variable per dataset, including dataset name, variable name, type, label, and format, and is treated as the authoritative catalog of the ADaM library. The actual ADaM analysis datasets (e.g., ADSL, ADAE, ADLB, ADTTE, ADRS) are also provided as Excel files and contain subject-level and domain-specific analysis data. Together, these two inputs provide enough structure and content to perform the required derivations, apply SAP-specified rules, and ensure traceability, without needing access to underlying raw or SDTM data.

Prompt snippet (Data foundation)

Use ADaM data provided as Excel files plus proc_contents.xlsx (SAS PROC CONTENTS export) as the sole analysis source. Do not request or rely on raw EDC data or SDTM; infer all concepts from ADaM + Protocol + SAP + mock shell.

High-Level Workflow

The workflow is intentionally simple and repeatable: it always operates on one TLF at a time and follows a fixed sequence of steps. First, it performs a brief intake and scoping of the requested display; second, it constructs a concept coverage index from the ADaM metadata in proc_contents.xlsx; third, it defines a targeted dataset and variable checklist for the TLF; fourth, it outlines a concise execution plan; and finally, it builds, quality-checks, and delivers the finished table or figure. This structure balances transparency, reproducibility, and the need for efficient iteration.

Prompt snippet (High-level workflow)

Work one TLF at a time with concise messages. Follow a fixed workflow: (1) Intake, (2) Concept Coverage Index from proc_contents.xlsx, (3) Targeted dataset checklist, (4) Execution plan, (5) Build → QC → Deliver.

Step 1 – Intake and Table Scoping

For each TLF, the process starts by clarifying what needs to be reproduced and how it is defined in the study documentation. The Protocol is reviewed to extract succinct study design information—phase, indication, randomization ratio, number and nature of treatment arms, blinding, and key visits or time points—which provides context for how treatment columns and analysis populations should behave. The SAP and mock shells are then used to identify population definitions (e.g., safety, ITT, per-protocol, biomarker subsets), endpoint definitions, confidence-interval methods, required statistics, and, importantly, the exact treatment column layout: arm labels, pooled groups, special columns, and their order. This step ends with a clear narrative description of the population, endpoints, treatment columns, and structure of the TLF.

Prompt snippet (Intake & scoping)

Step 1 – Intake: Ask me which TLF to replicate (e.g., baseline → safety → efficacy). From the Protocol, extract study design (phase, randomization, treatment arms, blinding, key visits). From the SAP and Mock Shell, extract population definitions and the exact treatment column structure (arm labels, pooled groups, column order, and any special columns). Confirm which files you received (Protocol, SAP, Mock, proc_contents.xlsx, ADaM Excel data) and briefly state any obvious gaps.

Step 2 – Concept Coverage Index (ADaM Only)

Next, the system builds a concept coverage index using only the PROC CONTENTS metadata for the ADaM library. It automatically detects the dataset-name column in proc_contents.xlsx (for example, DATASET, MEMNAME, or DOMAIN) and treats the unique values as the canonical list of ADaM datasets. For each dataset, all variables are

catalogued along with their labels, types, and formats, and their likely roles are inferred (identifiers, treatment variables, visit and time variables, analysis parameters, values, flags, and time-to-event endpoints or censoring indicators). For the current TLF, each needed concept—baseline values, changes, adverse-event categories, time-to-event endpoints, responder flags, and so on—is mapped to the datasets and variables where it appears. If a concept exists in multiple ADaM datasets or versions, all relevant sources are brought into scope and later reconciled via joining or stacking.

Prompt snippet (Concept Coverage Index)

Step 2 – Concept Coverage Index: In proc_contents.xlsx, auto-detect the ADaM dataset name column, treat the unique values as the canonical ADaM list, and build an index of all variables. For the chosen TLF, map each required concept to all ADaM datasets/variables where it appears.

Step 3 – Targeted Dataset Checklist

Using the concept coverage index plus the SAP and mock shell, the system creates a concise, TLF-specific dataset checklist. Rather than pulling everything, it identifies exactly which ADaM datasets are needed (for example, ADSL with ADAE for safety, or ADSL with ADLB for laboratory summaries) and which key variables are required to construct the analysis population, endpoints, stratifications, and treatment columns. At the same time, it defines minimal join keys—typically STUDYID and USUBJID, with PARAMCD, AVISITN, ASEQ, or analysis date where necessary—to keep joins interpretable and reproducible. Any clearly missing dataset or variable needed to satisfy the SAP and mock is flagged at this stage as a potential gap.

Prompt snippet (Targeted dataset checklist)

Step 3 – Targeted Dataset Checklist: From the SAP/mock derive populations, windows, endpoints, units, CI methods, and treatment column layout. Using the ADaM concept index, list ≤5 bullets with exact ADaM dataset names, critical variables, and minimal join keys needed for this TLF.

Step 4 – Execution Plan

Before performing any calculations, the system briefly summarizes how it will build the TLF. This includes how the analysis population and denominators (Big N and small n) will be derived using ADaM flags such as SAFFL and EFFFL, how treatment columns and pools will be constructed based on design and SAP, which ADaM datasets will be joined or stacked using the previously defined keys, and which statistics and windows will be applied to each parameter or endpoint. The execution plan is intentionally short, but it gives a clear, auditable bridge from inputs (documents and ADaM) to the final layout.

Prompt snippet (Execution plan)

Step 4 – Execution Plan: In 1–2 sentences, summarize the plan for this TLF: analysis population and denominators, which ADaM datasets will be joined, keys used, and key statistics. Proceed based on SAP/mock unless a blocking ambiguity exists.

Step 5 – Build → QC → Deliver

In the build phase, the system standardizes variable names, labels, formats, and codes across the selected ADaM datasets, then performs the necessary joins and stacks using the agreed keys. It applies the derivations specified in the SAP—such as defining baseline, changes from baseline, responder status, analysis windows, and censoring rules—and computes the requested descriptive or time-to-event statistics, including proportions and confidence intervals. Formatting logic is applied so that decimal precision, rounding, ordering of categories and treatment columns, and the exact wording of titles and footnotes match the mock shell. The QC component then verifies internal consistency: Big N and small n across related tables, totals and percentages, and alignment of methods and precision with the SAP. It also documents the lineage from each table cell back to specific ADaM datasets and variables, and records any assumptions made. If something cannot be produced because a dataset, variable, or rule

is missing or ambiguous, a short gap report is generated describing the issue and what would be required to resolve it. Finally, the completed TLF is exported as a fully formatted Excel worksheet (.xlsx) that mirrors the mock and is also printed in text form for easy review and archiving.

Prompt snippet (Build → QC → Deliver)

Step 5 – Build → QC → Deliver: Using ADaM Excel data, standardize variables, join/stack all datasets holding the same concepts, resolve duplicates per SAP, compute statistics with correct precision and treatment column order, and reproduce titles/footnotes exactly. Export the final TLF to Excel (.xlsx) and also print the table in the prompt.

Figures

When the SAP and mock specify figures instead of tables, the same ADaM-based derivation process is used to construct the underlying analysis dataset, and only the final presentation differs. The engine generates figures in PNG format, choosing a canvas size that ensures no content is cut off and that axis labels, tick marks, legends, and annotations do not overlap. Visual settings are aligned with the mock shell specifications so that figures can be dropped directly into clinical study reports or presentations alongside the tabular outputs.

Prompt snippet (Figures)

For figures, derive the analysis dataset from ADaM in the same way as tables. Generate a PNG figure with no overlapping labels, using an appropriate canvas size so that all content, legends, and axes are clearly visible and consistent with the mock.

Self-Check and Governance

As a final safeguard, the system performs a self-check to ensure that all ADaM datasets containing a required concept have been considered, that population counts and percentages are consistent across related TLFs sharing the same analysis population, and that confidence interval methods, decimal precision, ordering, treatment column layout, titles, and footnotes all match the SAP and mock shells exactly. This governance layer, combined with transparent documentation of assumptions and gaps, is what makes the ADaM in Excel approach robust enough for use in submission outputs.

Prompt snippet (Self-check & governance)

Before finalizing any TLF, perform a self-check: confirm that all ADaM datasets holding each required concept were used, Big N/small n match across related TLFs, treatment columns match the SAP/mock, CI methods and decimal places follow the SAP, and titles/footnotes exactly match the mock.
