PharmaSUG 2026 - Paper 312

# Trust but Verify: How ChatGPT and SAS Can Be Comrades

Steve Black MSPH, Neurocrine Biostatistics Inc.

## ABSTRACT

SAS® programmers in clinical development are under constant pressure to move faster while upholding uncompromising standards for quality and compliance. ChatGPT introduces a secure, governed large-language-model platform that can help navigate this tension when used thoughtfully. This paper presents practical, real-world examples of how ChatGPT has been applied to SAS programming workflows, including SDTM and ADaM production, derivation logic clarification, TFL output creation, and validation efforts. Particular emphasis is placed on its use within regulated environments, reinforcing reproducibility, traceability, and the continuing necessity of sound programmer judgment. Rather than automating decisions or replacing established roles, ChatGPT is positioned as a trusted comrade—useful, disciplined, and effective—working alongside SAS programmers without taking anyone's job hostage.

## INTRODUCTION

The rapid emergence of large language models has sparked both excitement and concern within regulated clinical programming environments. For SAS programmers, the promise of increased efficiency is often met with understandable skepticism around reproducibility, validation, and the role of human oversight. In an environment governed by strict regulatory expectations, any new technology must prove not only that it can accelerate work, but that it can do so without compromising control, accountability, or quality.

ChatGPT represents a new class of assistive technology that, when used appropriately, can support experienced SAS programmers rather than replace them. Its value lies not in making decisions, but in accelerating routine tasks, clarifying logic, and supporting consistency across deliverables. When paired with established SAS workflows and validation practices, ChatGPT can function as a reliable collaborator—operating under explicit direction and continuous verification.

This paper explores practical, real-world use cases demonstrating how ChatGPT can be integrated into SAS programming activities such as SDTM and ADaM production, TFL output creation, and validation efforts. Throughout these examples, the emphasis remains clear: trust, but verify.

## FRAMING THE ROLE OF CHATGPT

Before diving into examples, it is important to define what ChatGPT is—and what it is not—within a regulated SAS programming environment. ChatGPT is not an authority, a validator, or a decision-maker. It does not replace specifications, SOPs, or programmer accountability.

Instead, it functions as an assistive collaborator that can help draft logic, clarify requirements, and explore implementation approaches under explicit direction. All outputs must still be reviewed, verified, and implemented by the programmer within established development and validation workflows.

A useful analogy is to think of ChatGPT as a very smart intern—capable and efficient, but requiring clear instruction and consistent verification. It can move quickly, it can suggest ideas, and it can draft solid first passes. But it does not own the deliverable. The programmer does.

## SDTM PRODUCTION SUPPORT

When working with SDTM specifications, I have prompted ChatGPT with instructions such as:

"Here is my specification document and an example program. Create a first pass at this domain using the program as a template. Ask me questions as you proceed."

The results have been mixed—in a good way. Sometimes it produces strong, structured logic and walks through each variable carefully using both the template and its own interpretation. Other times the response is more generic and requires refinement. In either case, it provides a useful starting point.

I have also used ChatGPT to review specifications alongside programs to identify inconsistencies and suggest updates to define comments. It can help spot areas where documentation and implementation drift apart. Comparing specifications across domains to maintain style consistency is another area where it performs well.

Adding comments is probably the easiest win. I can paste in my code, ask for comments in my style, and it turns around a clean, readable version in seconds. Providing code and asking for comments in a specific style consistently yields clean, thorough results. This alone saves a lot of time and improves readability for both myself and anyone else who needs to take a peek into the program.

## ADAM DATASET DEVELOPMENT

In ADaM development, ChatGPT has been helpful in reviewing derivation logic and identifying areas where specifications could be clearer. By providing the ADaM specification and the implemented logic, I can ask it to highlight gaps, ambiguities, or opportunities to strengthen documentation.

When reasoning through complex analysis flags or endpoint logic, I provide the SAP, protocol details, and SDTM data structure, including variable summaries. The output often serves as a structured starting point. In some cases, it suggests additional logic or identifies edge cases I may not have initially considered.

For specification development involving multiple treatment periods, I have used ChatGPT to expand a single row into the necessary additional rows, adjusting values appropriately. This improves text consistency and reduces typographical errors, which are particularly helpful when working in large specification spreadsheets.

Performance tuning is another practical use. By sharing portions of a SAS log, I can ask ChatGPT to identify processes contributing to longer runtimes. Sometimes the recommendation is simply confirmation that the process is necessary. Other times, removing an unnecessary sort or restructuring a step results in noticeable improvement.

## TFL OUTPUT CREATION

For TFL development, I provide the shell, a template program, and dataset contents, then ask ChatGPT to draft code aligned with our macro framework. The first pass is rarely final, but with clarification prompts, it often gets most of the way there quickly.

I frequently use it for structured comment insertion and formatting updates. It handles these repetitive tasks efficiently and consistently.

In batch processing scenarios, I have used screenshots of outputs and program indices to confirm expected deliverables were generated. While this can certainly be done in SAS, it is sometimes convenient to delegate the comparison task and receive a structured summary.

Code efficiency reviews have also produced useful results. While not every suggestion is adopted, some have led to meaningful improvements.

## VALIDATION AND QC SUPPORT

During macro development, I have asked ChatGPT to generate test cases and datasets designed to validate logic and expose potential failure points. This has strengthened robustness prior to formal validation.

When inheriting code written in a different style, I paste it into ChatGPT and request a summary explanation. This provides a faster starting point for understanding structure and intent before making updates.

As always, these outputs are reviewed carefully. ChatGPT supports the thinking process, it does not replace it.

## LESSONS LEARNED AND GUARDRAILS

ChatGPT performs particularly well at reviewing documents, summarizing code, and identifying structural inconsistencies. Its understanding of SAS and macro language continues to improve, though occasional corrections are still necessary.

Everything it produces requires verification. Programs must be executed. Outputs must be reviewed. Documentation must be cross-checked against source materials. Direct acceptance without review is not appropriate in a regulated environment.

Prompt design matters. Framing instructions clearly - for example, specifying that assumptions should not be fabricated and that clarifying questions should be asked - consistently improves output quality.

Will programmers be replaced? Unlikely. However, as AI tools improve efficiency, expectations will increase. Managing more studies with the same number of programmers may become standard. The role evolves, but it does not disappear.

## CONCLUSION

Large language models are not going away, and ignoring them will not make them less relevant to our work. In regulated clinical programming, the question is not whether ChatGPT can write SAS code—it can—but whether it can be used responsibly within established standards and validation frameworks. Based on the examples presented, the answer is yes, when applied with discipline and clear boundaries.

ChatGPT performs best when treated as a knowledgeable but supervised assistant. It can draft logic, surface inconsistencies, suggest efficiencies, and accelerate documentation. But it does not replace specifications, it does not understand regulatory accountability, and it does not sign off on outputs. That responsibility remains firmly with the programmer.

The principle remains simple: trust but verify. Every suggestion must be reviewed. Every program must be executed. Every output must be validated.

Used this way, ChatGPT becomes less of a threat and more of a comrade—working side by side with the programmer, responding to direction, and improving through clearer prompts and stronger communication. It will not take your job hostage. It will change how you work. And in a regulated world, disciplined collaboration may be exactly the kind of comrade we need.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Steve Black
Neurocrine Biosciences
steven.c.black@gmail.com