Paper ET01

AI as Copilot: Empowering, Not Replacing, Clinical Statistical Programmers

Bhavesh Patkar, Syneos Health, Hyderabad, India
Brian Thornton, Spaulding Clinical Research, West Bend, Wisconsin, USA

ABSTRACT

The rapid rise of artificial intelligence (AI) and large language models has sparked debate about whether programmers will be replaced by machines. In reality, AI is not a substitute for human expertise but a powerful copilot that can augment productivity, enhance quality, and reduce repetitive work. This paper explores how AI tools can support clinical statistical programmers in clinical research by assisting with knowledge retrieval, code generation for well-defined tasks, pre-validation quality control, and program understanding and reuse. Rather than displacing programmers, AI enables clinical statistical programmers to focus on higher-value activities such as complex analyses, problem solving, and ensuring compliance with regulatory standards. Practical examples demonstrate how AI can be integrated responsibly into clinical statistical programming workflows, highlighting both benefits and limitations. Readers will gain actionable strategies to position AI as a trusted assistant that accelerates delivery while preserving the rigor and judgment essential to high-quality clinical statistical programming.

INTRODUCTION

The rapid advancement of artificial intelligence (AI), particularly large language models (LLMs), has intensified discussion about the future of technical roles across many industries. Within clinical research, statistical programmers are increasingly exposed to AI-enabled tools that can generate code, explain logic, and suggest optimizations in real time. While these capabilities offer clear efficiency gains, they have also raised concerns that automation may eventually replace the need for skilled programmers.

In regulated clinical development environments, however, the role of the statistical programmer extends far beyond writing syntactically correct code. Programmers are responsible for implementing complex study designs, interpreting analysis specifications, ensuring traceability, and producing outputs that withstand regulatory scrutiny. These responsibilities require contextual understanding, judgment, and accountability that cannot be delegated to automated tools.

This paper argues that AI is most effective when positioned not as a replacement for clinical statistical programmers, but as a copilot that supports their work. When used responsibly, AI can reduce time spent on repetitive or low-value tasks, improve code quality through early feedback, and help programmers focus on higher-order activities that directly contribute to study integrity and delivery. The following sections examine the role of clinical statistical programmers in clinical research, identify realistic areas where AI can add value, and propose a practical framework for integrating AI into programming workflows while maintaining compliance and quality.

THE ROLE OF CLINICAL STATISTICAL PROGRAMMERS IN CLINICAL RESEARCH

Clinical statistical programmers play a critical role in transforming raw clinical trial data into structured, analysis-ready datasets and interpretable outputs. Their work underpins key study deliverables, including submission datasets, tables, listings, and figures, all of which must meet strict regulatory and quality standards.

Unlike many general software development contexts, clinical statistical programming is governed by detailed specifications, standard data models, and evolving regulatory expectations. Programmers must interpret analysis plans, apply controlled terminology, manage complex derivations, and ensure consistency across datasets and outputs. Decisions made during programming often require nuanced judgment, particularly when specifications are ambiguous, data are incomplete, or edge cases arise.

Accountability is another defining characteristic of the statistical programming role. Programmers are expected to validate their work, participate in quality control processes, and support audits or regulatory questions long after deliverables are produced. While tools can assist with implementation, responsibility for the correctness, traceability, and interpretability of results remains firmly with the human programmer.

These characteristics make clinical statistical programming fundamentally different from environments where automated code generation can be adopted with minimal oversight. Any introduction of AI into this space must therefore respect the central role of human expertise and ensure that accountability is preserved.

AI AS A COPILOT IN CLINICAL STATISTICAL PROGRAMMING

When positioned as a copilot, AI serves as an assistive tool that supports programmers in well-defined tasks while leaving decision-making and responsibility with the user. In this role, AI does not replace existing development or quality processes but augments them by providing rapid feedback and reducing manual effort.

SUPPORTED TASKS AND USE CASES

AI tools can add value across a range of clinical statistical programming activities when applied to well-defined, bounded tasks. Rather than replacing existing development, QC, or validation processes, AI is most effective when used to support programmers with information retrieval, structured code assistance, logic comprehension, and early-stage review.

Across the programming lifecycle, AI can be used as a knowledge assistant, a drafting aid for repetitive or structured code, a helper for reasoning through complex logic, and a support mechanism for pre-validation self-review. These uses span both simple and more complex programming tasks, with the level of required human oversight increasing as task complexity and risk increase.

The following section presents practical use cases that illustrate how these forms of AI assistance can be applied responsibly in clinical statistical programming workflows. In all instances, AI-generated output is treated as draft material and remains subject to full human review, testing, and validation before use in regulated environments.

HUMAN OVERSIGHT AND ACCOUNTABILITY

Across all use cases, it is essential that AI-generated output is treated as draft material. Clinical statistical programmers must review, test, and validate all AI-assisted code before it is incorporated into production workflows. AI tools do not have visibility into full study context, organisational standards, or regulatory intent, and therefore cannot independently ensure compliance or correctness.

Maintaining this distinction between assistance and authority is key to using AI safely and effectively. By reinforcing the programmer's role as the final decision-maker, organisations can benefit from AI-driven efficiencies without compromising quality or accountability.

PRACTICAL USE CASES: AI AS A COPILOT IN CLINICAL STATISTICAL PROGRAMMING

The following use cases illustrate practical examples of how AI can be used as a copilot to support clinical statistical programmers in clinical research. The use cases are ordered from the most straightforward and low-risk applications to more complex scenarios that require greater programmer judgment and experience. Together, they demonstrate how AI can progressively assist programmers while preserving human accountability at every stage.

USE CASE 1: AI AS A KNOWLEDGE ASSISTANT FOR SAS® PROCEDURES AND SYNTAX

AI can be used as an interactive knowledge assistant to help clinical statistical programmers identify appropriate SAS® procedures, functions, and options for a given analytical task. Rather than manually searching reference manuals, online documentation, or internal knowledge bases, programmers can pose natural-language questions describing their requirements.

In response, AI can provide high-level explanations, suggest relevant procedures or functions, and highlight commonly used options or considerations. The programmer then evaluates these suggestions, confirms details against official SAS documentation, and applies domain-specific and regulatory judgment before implementation. In this role, AI reduces time spent searching static resources while preserving the authority of official documentation and the expertise of the programmer.

USE CASE 2: GENERATING REPETITIVE CODE FROM EXISTING STUDY DOCUMENTATION

AI can be used to assist with generating repetitive SAS® code from existing study documentation, such as case report forms (CRFs), questionnaires, or specification tables. Clinical statistical programming often requires translating coded values into descriptive text through formats or conditional logic, a task that can involve extensive manual typing and cut-and-paste activities.

By providing the AI tool with structured documentation containing codes and corresponding labels, programmers can request draft PROC FORMAT definitions or equivalent conditional statements. The generated code is then reviewed for accuracy, verified against the source documentation, and incorporated into production programs following standard review procedures. This use case reduces manual effort and transcription errors while maintaining full human oversight.

USE CASE 3: AI-ASSISTED GENERATION OF COMPLEX DATA STEP LOGIC

AI can support the creation of complex SAS DATA step logic for structured transformations that are time-consuming to write from scratch. For example, when a single source record contains information applicable to multiple visits or study periods, programmers may need to expand the data into multiple analysis-ready observations.

In this scenario, AI can be used to draft DATA step logic that parses text values, applies looping constructs, and outputs multiple records according to defined rules. The programmer reviews and adapts the suggested code to the specific dataset structure and validates the results through testing and quality control checks. AI accelerates development of localized code blocks while leaving design decisions and validation with the programmer.

USE CASE 4: AI SUPPORT FOR DEVELOPING COMPLEX CODE BLOCKS (E.G., MANY-TO-MANY MERGES WITH PROC SQL)

AI can be used to assist clinical statistical programmers in developing complex SAS® code blocks that require careful reasoning and are prone to error if implemented incorrectly. Many-to-many merges using PROC SQL are a common example of such complexity in clinical statistical programming.

In this use case, programmers can use AI to help reason through the structure of a many-to-many merge by describing the source datasets, key variables, and intended outcomes. AI can provide conceptual explanations of join behavior, discuss potential risks such as record inflation or unintended duplication, and outline the PROC SQL options relevant to the scenario.

AI can also be used to draft an initial PROC SQL code block and suggest supporting pre- and post-merge checks, such as expected record counts, duplicate key reviews, and sample inspections. The programmer reviews and adapts the suggested code, implements the merge, and performs all required testing and validation. In this role, AI accelerates development and understanding of complex code blocks while leaving full responsibility for correctness and approval with the statistical programmer.

USE CASE 5: AI-SUPPORTED PRE-VALIDATION QC AND SELF-REVIEW

AI can be used to support programmers during the pre-validation quality control (QC) phase of development. After completing an initial version of a program, programmers can use AI as a review assistant to examine code step by step and identify potential issues.

In this role, AI can help highlight logic that may not handle all data scenarios, undocumented assumptions, missing edge case handling, or sections of code that could behave unexpectedly. AI does not execute programs or validate results; instead, it prompts the programmer to reassess logic paths, review merge conditions, and confirm alignment with specifications. This supports earlier detection of issues before programs are released for independent validation.

USE CASE 6: AI-ASSISTED UNDERSTANDING, DOCUMENTATION, AND REUSE OF INHERITED PROGRAMS

AI can be used to assist clinical statistical programmers who inherit complex SAS® programs written by previous authors who are no longer available. In such situations, programmers can use AI to generate plain-language walkthroughs and logical maps of program processing steps, helping accelerate understanding of overall intent and data flow.

Based on this understanding, AI can assist in drafting clear and consistent comment text to document program logic, assumptions, and dependencies. When updates are required, AI can help identify where changes should be made and explain potential downstream impacts. With improved understanding and documentation, the program can then serve as a reliable starting point for a new study, enabling safe reuse while preserving human judgment, review, and accountability throughout the process.

LIMITATIONS, RISKS, AND GOVERNANCE

While AI can provide meaningful support across the use cases described above, its application in clinical statistical programming must be carefully bounded. Clinical trial programming operates within a regulated environment where correctness, traceability, and accountability are paramount. As such, AI must be treated as an assistive tool rather than an authoritative source of truth.

One key limitation of AI is the potential to generate output that appears plausible but is logically incorrect or incomplete. AI-generated code may omit edge cases, mishandle missing data, or make assumptions that are not aligned with study specifications or regulatory expectations. For this reason, all AI-assisted outputs must be reviewed, tested, and validated by a qualified statistical programmer before use.

Data privacy and governance represent additional considerations. Clinical statistical programming often involves sensitive or proprietary information that may not be appropriate to share with external AI tools. Organisations should establish clear policies defining what information can be provided to AI systems and ensure that approved tools comply with internal security and data protection standards.

Another important risk is overreliance on AI suggestions. As use cases increase in complexity, the likelihood of subtle errors also increases. Programmers must remain actively engaged in reasoning through code logic, particularly for complex transformations, merges, and derivations. AI should support understanding and efficiency, not replace critical thinking or domain expertise.

To address these risks, organisations should implement governance practices that define acceptable AI use. Such practices may include restricting AI use to specific tasks, requiring explicit human review of all AI-assisted outputs, documenting AI involvement where appropriate, and prohibiting the use of AI for activities such as independent validation or final approval. By establishing clear guardrails, teams can safely realise the benefits of AI while maintaining compliance, quality, and accountability.

RESPONSIBLE INTEGRATION FRAMEWORK

To operationalise the safe and effective use of AI as a copilot, organisations can adopt a simple integration framework that aligns AI use with task complexity and risk. This framework emphasises human oversight, transparency, and clear separation of responsibilities.

Approved AI use cases should be clearly defined. Low-risk activities, such as using AI as a knowledge assistant or generating repetitive code from existing documentation, are generally suitable entry points. More complex use cases, including assistance with complex code blocks or inherited program understanding, may be permitted with additional review and testing requirements.

Human review is mandatory for all AI-assisted outputs. AI-generated code, explanations, or documentation must be treated as draft material. Programmers are responsible for reviewing logic, confirming alignment with specifications, executing appropriate tests, and documenting any changes made as a result of AI assistance.

Clear boundaries should be established for prohibited uses. AI should not be used to perform independent validation, approve deliverables, or make final determinations regarding regulatory compliance. These activities require independent human judgment and remain outside the appropriate scope of AI assistance.

Transparency and documentation support auditability. Where AI is used in a meaningful way, teams may document its role at a high level, such as noting that AI was used to assist with code review or documentation drafting. This transparency reinforces accountability without overstating AI’s role.

By aligning AI use with clearly defined tasks, enforcing human review, and maintaining strong governance, organisations can integrate AI responsibly into statistical programming workflows. This approach enables programmers to benefit from increased efficiency and support while preserving the rigor, judgment, and accountability required in clinical research.

FUTURE OUTLOOK

As AI tools continue to mature, their role in statistical programming is likely to expand in depth rather than breadth. Improvements in model reliability, explainability, and enterprise governance will make AI assistance more predictable and easier to integrate into regulated workflows. However, the fundamental need for human judgment, contextual understanding, and accountability will remain unchanged.

In the near term, adoption is expected to focus on areas that are already well understood and procedurally defined, such as code comprehension, documentation support, and generation of localized code blocks. These uses align naturally with existing programming practices and can be incorporated incrementally without disrupting established validation models.

Over time, organisations may develop more formal standards and tooling to support AI-assisted development, including approved prompts, reusable templates, and audit-friendly documentation practices. Rather than reducing the importance of clinical statistical programmers, these developments are likely to elevate the role by shifting effort away from repetitive tasks and toward activities that require interpretation, problem solving, and regulatory awareness.

Ultimately, AI is unlikely to replace clinical statistical programmers in clinical research. Instead, it will increasingly serve as a copilot that enhances productivity and quality when applied thoughtfully, governed appropriately, and guided by experienced professionals.

CONCLUSION

The rapid emergence of AI has prompted understandable questions about its impact on the role of clinical statistical programmers in clinical research. This paper has argued that AI is best viewed not as a replacement for human expertise, but as a copilot that supports programmers by reducing repetitive effort, accelerating understanding, and assisting with well-defined technical tasks.

Through a progression of practical use cases, this paper has demonstrated how AI can be applied responsibly across the programming lifecycle—from serving as a knowledge assistant and generating repetitive code, to supporting complex logic development, pre-validation quality control, and the understanding and reuse of inherited programs. In each example, AI provides value by augmenting human capability while preserving accountability, judgment, and regulatory responsibility.

Crucially, effective use of AI depends on strong governance and disciplined integration. Clear boundaries around acceptable use, mandatory human review, and explicit exclusion of AI from validation and approval activities are essential to maintaining quality and compliance in regulated environments.

When applied within these guardrails, AI can empower clinical statistical programmers to focus more time and attention on higher-value activities such as interpretation, problem solving, and ensuring the scientific integrity of clinical trial results. In this role, AI does not diminish the importance of the statistical programmer—it reinforces their central role in delivering reliable, high-quality clinical research.

AI USE DISCLOSURE

AI tools were used to assist with drafting and refining portions of this paper, including improving clarity, structure, and technical explanations. All content was reviewed, edited, and validated by the authors, who retain full responsibility for the accuracy and integrity of the material presented.

REFERENCES

OpenAI. (2025). ChatGPT (GPT-4o, 2 July version) [Large language model]. https://chat.openai.com/
SAS Institute Inc. SAS® 9.4 Documentation. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Bhavesh S. Patkar
Company: Syneos Health
Email: bhavesh.patkar@gmail.com

Author Name: Brian Thornton
Company: Spaulding Clinical
Email: brian.thornton@spauldingclinical.com

DISCLAIMER

The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of their respective organizations.