From SAS to Cursor: Vibe-Coding into SAS, R & Python

Kevin Lee, ClinVia, Philadelphia, PA

ABSTRACT

The rise of Gen AI is revolutionizing how coding is performed across industry, and “vibe coding” stands at the forefront of this transformation. Coined from Andrej Karpathy’s idea of “embracing the vibes” of AI-assisted coding, vibe coding represents a seamless flow between human logic and AI coding agents—where programmers prompt, review, and collaborate with AI to produce code efficiently and intelligently. As an example, Microsoft CEO Satya Nadella estimated in April 2025 that 20% to 30% of Microsoft's code was generated by AI.

The presentation explores how vibe coding will reshape programming in Biometrics, where SAS, R, and Python remain essential. It illustrates how conversational AI tools (ChatGPT, Gemini, Claude), AI-native IDEs (Cursor, Windsurf, GitHub Copilot) and customized AI Coding Agents and Agentic workflow enhance traditional workflow - automating coding, debugging, and validation processes while preserving scientific and regulatory rigor.

The talk introduces customized Vibe Coding Agents and Agentic workflow built for Biometrics, integrating CDISC, ADaM, and TLF standards with GxP-compliant validation frameworks. Real-world examples demonstrate how these systems accelerate programming cycles by 25–40%, improve documentation, and lower technical barriers across languages in SAS, R and Python.

While the benefits are profound such as productivity gains, democratization of coding, and cross-functional collaboration, the presentation also addresses risks such as AI hallucination, compliance, and over-reliance without human oversight.

Finally, it offers a forward-looking view of the “AI-augmented biometrics team,” where statistical programmers evolve into AI managers and collaborators, driving innovation and quality in clinical research and development.

Introduction of Vibe-coding

Vibe-coding represents a transformative shift in codes development, where programmers leverage AI, particularly large language models (LLMs), to generate code through natural language prompts rather than manual syntax-driven writing. The term was introduced by Andrej Karpathy, co-founder of OpenAI, in February 2025.

In his widely shared statement, Karpathy described it as "a new kind of coding I call 'vibe coding,' where you fully give in to the vibes, embrace exponentials, and forget that the code even exists."

This approach emphasizes a seamless flow or "vibe" between human intent and AI execution: programmers articulate high-level logic or desired outcomes in conversational prompts, review and refine the generated code, and iterate as needed. Rather than focusing on low-level syntax details, the coder acts as a director, guiding AI agents to produce functional codes. This paradigm has gained significant traction due to rapid advancements in AI tools, enabling faster prototyping and reducing the cognitive load of traditional coding.

Current Coding in Biometrics: SAS, R and Python

In biometrics and statistical programming, three primary languages dominate workflows, each serving distinct yet overlapping roles in regulatory-compliant data analysis.

- SAS remains the regulatory backbone, particularly for generating standardized datasets such as SDTM and ADaM, and TFLs. Its strength lies in traceability, reproducibility, and long-standing acceptance by regulatory authorities like the FDA.

- R excels in advanced data visualization, interactive applications via Shiny, and increasingly in SDTM/ADaM/TFL production. It offers flexibility for exploratory analysis and modern statistical techniques.

- Python supports data pipelines, automation, AI/ML integration, and utility functions. It is gaining ground for handling large-scale data processing and emerging AI-driven tasks.

This reflects the specialized needs of biometrics: compliance and validation in regulated environments, alongside innovation in analytics and visualization.

The integration of Vibe coding in Biometrics

Vibe-coding can integrate into biometrics by lowering barriers across SAS, R, and Python, enabling statisticians and programmers to focus on scientific logic rather than language-specific syntax. Tools like ChatGPT initially demonstrated this potential through simple prompts that generate equivalent code in multiple languages for tasks like the prompts below.

- Code Development in R
  - Prompt:

    Write a **R script** that groups data by "trta" and "avisitn" columns in "advs.xpt"
    and calculates the mean of the "aval" column for each group at parmcd ='WEIGHT'

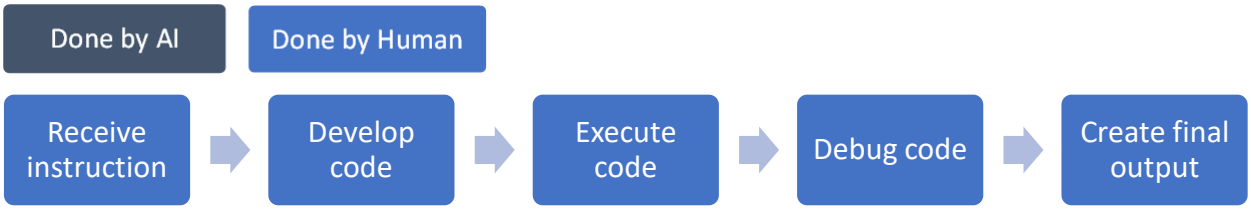
  - Output:

    ```R
    # Load necessary libraries
    library(haven)
    library(dplyr)

    # Read the XPT file
    advs <- read_xpt("path/to/advs.xpt")

    # Filter the dataset for parmcd == 'WEIGHT'
    advs_weight <- advs %>% filter(parmcd == 'WEIGHT')

    # Group by 'trta' and 'avisitn' and calculate the mean of 'aval' for each group
    mean_aval <- advs_weight %>%
      group_by(trta, avisitn) %>%
      summarise(mean_aval = mean(aval, na.rm = TRUE))

    # Print the result
    print(mean_aval)
    ```

And, SAS and python codes could be developed in the exact same prompt just replacing R with SAS or Python. This cross-language fluidity democratizes programming, allowing teams to transition more easily and maintain consistency.

Programming Workflow Before ChatGPT

Before the advent of ChatGPT, the traditional programming workflow followed a linear, manual process. A programmer first received instructions or requirements, then translated them into code based on their own knowledge and documentation. The code was executed to test whether it worked as expected, often revealing errors that required careful debugging. This cycle of execution and debugging could repeat multiple times until the desired behavior was achieved. Only after resolving issues and refining the logic would the developer produce the final output, making the process time-consuming and highly dependent on individual expertise and experience.



Programming workflow after ChatGPT

With the introduction of ChatGPT, the programming workflow has become more interactive and efficient. After receiving instructions, programmers can collaborate with ChatGPT to rapidly develop code, explore alternative implementations, and clarify requirements in natural language. ChatGPT can also help execute reasoning about the code and identify bugs, suggesting fixes and optimizations in real time. This reduces the repetitive trial-and-error cycle and shortens development time, allowing programmers to focus more on problem-solving, system design, and final validation before delivering the final output.

Introduction of AI Agents

An AI agent is an autonomous software system capable of interacting with its environment, collecting relevant data, and independently performing tasks while making decisions to achieve predefined goals. Unlike traditional large language models (LLMs) that primarily generate responses based on prompts, AI agents extend this capability by incorporating reasoning, planning, and action execution in dynamic, multi-step workflows. In the context of vibe-coding and statistical programming in biometrics, these agents represent a significant evolution, enabling more seamless and efficient code generation, execution, and iteration.

The core components of an AI agent include:

- LLM (e.g., advanced models such as ChatGPT-5.2 or Claude-4.5) - the central reasoning engine
- Prompts – instruction of AI Agent that define the agent's role, behavior, and procedural step.

  For example, AI Agent will be defined as
  'You are stat programmer.
  Use the input and generate SAS output.

  Follow below steps:
  1. Generate SAS codes based on input
  2. Execute generated SAS codes using SAS Studio.
  3. If there are error messages, debug SAS codes.
  4. Rerun SAS codes to create the output.

  Do Step 2 to 4 until there are no error message. '

- Tools (e.g., SAS Studio, R Studio, Jupyter) - provide access to external environments and functionalities
- Memory - a short-term context for ongoing interactions, long-term storage for persistent knowledge, and Retrieval-Augmented Generation (RAG) techniques to reference prior validated SAS codes, project histories, or domain-specific standards.

Together, these elements allow AI agents to operate with greater autonomy and reliability, transforming repetitive programming tasks into intelligent, self-correcting processes while maintaining human oversight for critical validation in regulated Biometrics environments.

Programming workflow with AI Agents

With AI Agents, the programming workflow becomes largely autonomous and continuous. After receiving high-level goals or requirements, an AI agent can independently develop code, execute it in a controlled environment, detect errors, and debug issues without constant human intervention. The agent iteratively refines the solution by observing execution results, validating outputs against objectives, and making improvements until the desired outcome is achieved.

Now, one should ask "what is the problem of this workflow?"

Programming workflow with AI Agents in Human in a loop – Agentic Workflow

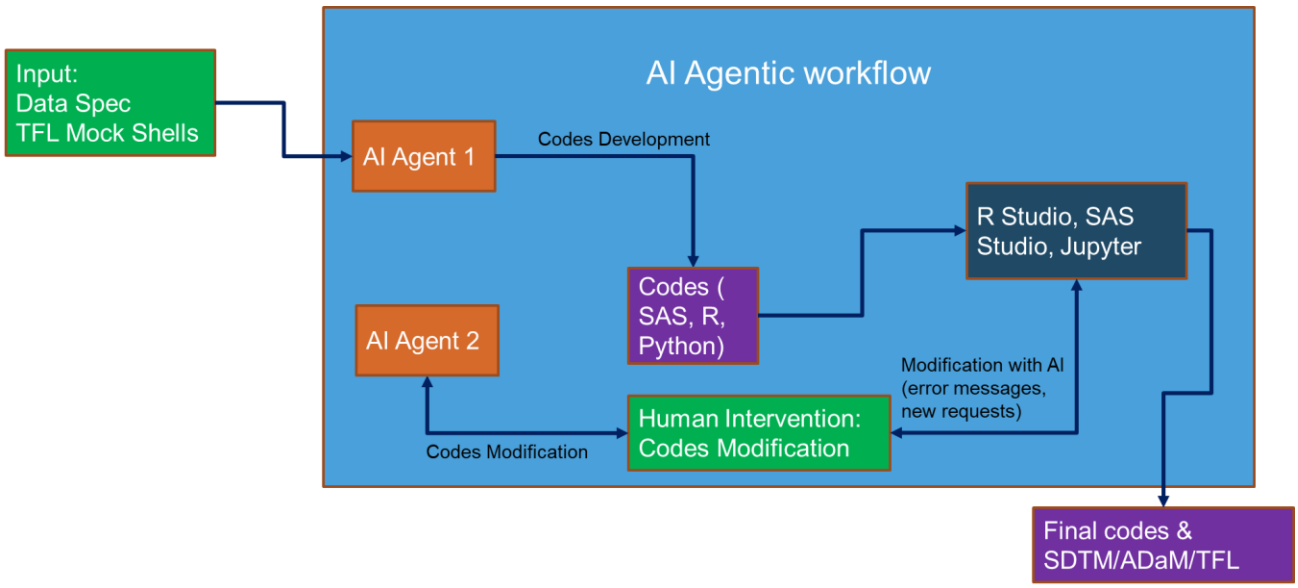
In a human-in-the-loop, agentic programming workflow, AI agents handle the core technical tasks—developing code, executing it, debugging errors, and generating final outputs while humans remain actively involved at critical control points. After instructions are received, the AI agent autonomously iterates through implementation and refinement, accelerating development and reducing manual effort. Crucially, humans validate the intermediate and final outputs to ensure correctness, reliability, compliance, and alignment with real-world requirements before delivery. This collaborative workflow balances the speed and scalability of AI agents with human judgment and accountability, mitigating risks while maximizing productivity and trust in the final software outcomes.

Vibe coding AI Agents in Industry - Cursor and Windsurf, github copilot

Industry-leading vibe-coding tools include:

- Cursor: An AI-native IDE designed for conversational programming. It supports seamless prompting within the editor, real-time code generation, and high acceptance of AI suggestions, aligning closely with Karpathy's "vibe" philosophy. It excels in rapid iteration and is popular for exploratory and prototype work.
- Windsurf: A context-aware, agent-powered IDE (formerly associated with advanced autocompletion) that emphasizes flow-state coding through features like autonomous agents (e.g., Cascade for multi-step reasoning) and real-time collaboration. It positions itself as an evolution toward fully agentic environments.
- GitHub Copilot: Provides inline suggestions across IDEs like VS Code, RStudio, and Jupyter, offering contextual completions and chat-based assistance for code generation and debugging.
- Claude Code – terminal-first AI coding assistant that understands and edits entire codebases, executes commands, and helps debug, refactor, and test code through natural language instructions

Customized AI coding agents for Biometrics



Benefits and Risks

Benefits include dramatic reductions in repetitive tasks, accelerating QC and development cycles by 25–40%, enabling seamless multi-language transitions (SAS/R/Python), democratizing access across experience levels, and improving documentation and traceability. Industry reports highlight efficiency gains of 10–60% for targeted tasks, with some developers saving over 10 hours weekly.

Risks encompass hallucinated or inaccurate code, limited domain understanding in generic tools, regulatory challenges (e.g., audit trails, GxP compliance, validation), data privacy concerns, and over-reliance on AI judgment, potentially compromising accountability.

Future of Biometrics in the era of Vibe-coding

Vibe-coding will fundamentally change Biometrics functions from manual, syntax-driven work to conversational, AI-augmented design of workflows and oversight. Roles will evolve from "coders" and "syntax experts" to "AI managers," subject matter experts, and collaborative partners in human-AI teams.

New skill sets include prompt engineering, proficiency with vibe-coding tools (e.g., Cursor, Windsurf, Copilot), validation of AI outputs, understanding agentic workflows, and human-centric abilities like critical thinking, creativity, EQ, and communication.

Collaboration policies delineate clear boundaries: AI as assistant (initial generation) or autonomous operator (routine tasks), with human oversight for accountability.

Looking ahead, biometrics teams can adopt vibe-coding strategies to become innovation hubs, bridging data, analytics, and AI while lowering language barriers and building "super teams" empowered by AI. Rather than replacing statistical programmers, biostatisticians and data scientists, vibe-coding empowers us to focus on high-value scientific and regulatory contributions, leading clinical development into an AI-augmented future.

REFERENCES

- ChatGPT in https://chatgpt.com/?ref=dotcom
- Wikipedia – Vibe Coding in https://en.wikipedia.org/wiki/Vibe_coding
- Wikipedia – AI Agent in https://en.wikipedia.org/wiki/AI_agent
- What are Agentic Workflows in https://weaviate.io/blog/what-are-agentic-workflows
- Cursor in https://cursor.com/
- Github Copilot in https://github.com/copilot

CONTACT INFORMATION

Your comments and questions are valued and welcomed. Please contact the author at

Kevin Lee
Clinvia LLC
kevin@clinvia.com

TRADEMARKS