PharmaSUG China 2025 - Paper AA-162

CodeGenius: An AI-Powered SAS Editor Enhancing Statistical Programming Efficiency through Intelligent Automation and Custom Integration

Yaohui Zhu, Xuejie Zhou, Jinling Li, Hao Chen, You Li, BeOne Medicines

ABSTRACT

In the rapidly evolving landscape of technological solutions for data, CodeGenius emerges as an innovative SAS editor specifically designed for Statistical Programming (SP).This innovative tool is engineered to enhance programming efficiency and intelligence by integrating advanced AI capabilities directly into the programming environment, thereby eliminating the need for external tools. Built on the robust Monaco Editor, CodeGenius offers a modern user experience similar to that of VS Code, facilitating a seamless transition from traditional SAS Enterprise Guide.

Key features of CodeGenius include AI-driven code automation, intelligent code completion, and strict adherence to company standards, ensuring consistency and maintainability across projects. The SPIS AI team's custom API enables sophisticated functions such as SAS code generation, formatting, and debugging, while a built-in chatbot supports dynamic user interactions. Additionally, CodeGenius leverages internal resources, such as SAS macros, to provide customized auto-completion, significantly enhancing code writing efficiency.

With its continuous enhancement capabilities, CodeGenius is poised to become an indispensable tool, streamlining programming processes and elevating the application of technological solutions in data. This positions CodeGenius as a critical platform for professionals seeking to optimize their statistical programming workflows.

INTRODUCTION

In today's pharmaceutical industry, statistical programming plays an important role in clinical trials. However, as the demands for compliance, efficiency, and data quality continue to rise, traditional SAS programming tools are increasingly revealing their limitations. In environments like SAS Enterprise Guide (EG), programmers often face tedious repetitive tasks, leading to low development efficiency and a higher risk of human error. Additionally, the industry demands higher standards for code standardization, maintainability, and compliance, yet existing tools offer limited capabilities in automation and intelligent assistance, making it difficult to meet the rapidly changing industry needs.

Simultaneously, the rapid advancement of AI technology presents unprecedented opportunities for transformation in clinical statistical programming. AI-driven features such as intelligent code automation, auto-completion, and standardization suggestions promise to significantly enhance programming efficiency, reduce repetitive work, and ensure code quality and compliance. However, generic AI editors often fall short of meeting the specific business processes and compliance standards of pharmaceutical companies, lacking industry-customized intelligent support.

Recognizing these challenges, the Scientific Programming (SP) team identified the need for an innovative solution to enhance programming efficiency and code quality. We developed CodeGenius—a smart SAS editor specifically designed for statistical programming in the pharmaceutical industry. CodeGenius not only integrates AI technology with internal resources like macro templates but also offers a modern interface based on Monaco Editor and powerful AI functionalities, providing an efficient, intelligent, and compliant programming environment. Through AI-driven code automation and intelligent code completion features, CodeGenius significantly boosts programmer productivity, reduces repetitive tasks, and enhances code consistency and quality.

This presentation will cover the main challenges faced, and how we have addressed these needs with CodeGenius as an innovative solution, helping pharmaceutical companies achieve breakthroughs in compliance, efficiency, and innovation. With this transformative tool, we aim to drive change in the field of

statistical programming, empowering programmers to focus on creative problem-solving and strategic development.

WHY WE CHOSE IN-HOUSE DEVELOPMENT

While AI-assisted coding tools like GitHub Copilot, Cursor, and Windsurf gain popularity, they fall critically short in addressing the specialized needs of statistical programming for clinical trials. Our analysis reveals four fundamental limitations in commercial solutions:

1. Domain expertise gaps present significant challenges. Generalized AI models often fail to comprehend SAS-specific structures, such as the syntax used in proc phreg survival analysis. Additionally, these models tend to overlook the nuances of clinical trials, including the inability to recognize CDISC variable naming conventions or ADaM metadata requirements.
2. Corporate knowledge isolation is another critical issue. There is zero access to proprietary assets, such as company-validated macros or standardized templates. Furthermore, security risks arise because external solutions cannot comply with clinical data governance policies.
3. Customization difficulties also pose problems. There are compatibility issues with existing systems, and an inability to meet specific needs. Additionally, security and compliance risks are a major concern. The improper use of external software can lead to data breaches, resulting in serious compliance issues.

Based on the reasons outlined above, we have decided to develop our own solution. As statistical programmers, we possess a deeper understanding of the specific business needs, allowing us to seamlessly switch roles between designer, developer, and user to ensure a well-rounded product. Embracing iterative progress and testing small, incremental ideas proves to be more effective than waiting for perfection.

SYSTEM ARCHITECTURE AND TECHNICAL IMPLEMENTATION

SYSTEM ARCHITECTURE

CodeGenius leverages a purpose-built three-tier architecture that balances cutting-edge AI capabilities with enterprise-grade security and domain-specific optimization. This foundation enables seamless integration of statistical programming expertise while maintaining GxP compliance (Figure 1).

Frontend Layer

Monaco Editor, developed by Microsoft, is a powerful open-source code editor and the core component of Visual Studio Code. It delivers a rich and seamless editing experience within web applications. With features such as syntax highlighting, IntelliSense, code folding, and more, Monaco Editor is well-suited for building sophisticated web-based development tools. Its high degree of customization allows easy integration into various web projects, providing a robust and versatile code editing interface.

In the CodeGenius project, Monaco Editor was chosen as the foundational SAS editor and tightly integrated with the Vue frontend framework. This close integration significantly enhances both the user experience and development efficiency, creating a smooth and responsive environment for SAS programming.

As the core engine behind VS Code, Monaco Editor offers an editing experience on par with Visual Studio Code itself, Through our custom development, we have implemented, but are not limited to, the following features:

1. Syntax Highlighting: Delivers context-aware syntax highlighting tailored for SAS programs, enabling developers to quickly differentiate between code segments like data steps and PROC SQL.

2. Multi-Tab Workspace: Supports multiple tabs, allowing users to effortlessly switch between different programs within the same interface.

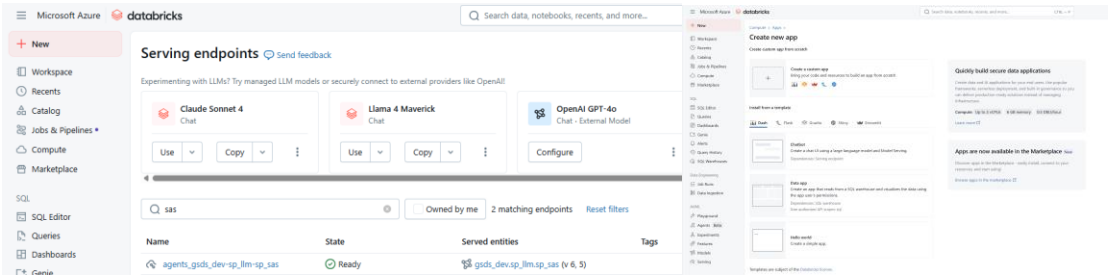
3. Keybinding Inheritance: Mimics SAS Enterprise Guide keybindings, easing the transition for users moving from SAS EG to Monaco Editor.

By leveraging Monaco Editor’s powerful features alongside Vue’s reactive framework, CodeGenius provides a modern, efficient, and user-friendly SAS coding environment that meets the demands.

AI Engine Layer

The AI Engine Layer is deployed on a dedicated Databricks platform, ensuring both security and stability. Its core components include:

1. Prompt Engineering API: Offers domain-optimized prompt templates that convert natural language inputs into SAS code.
2. Multi-Turn Dialogue Manager: The context-aware chatbot can maintain continuity across more than 10 conversational turns, providing a more intelligent interactive experience. At the same time, the system leverages Retrieval-Augmented Generation (RAG) technology to perform real-time retrieval of internal corporate metadata of standard macros and template code, and naming convention documents. By combining this information with the conversational context, it recommends the most relevant template code and macros to users, significantly enhancing code accuracy and reusability.
3. Support for Multiple Large Language Models: The system supports a variety of LLMs, each with its own strengths and use cases. Includes deepseek-r1, gpt41, gpt41-mini, gpt41-nano, claude-sonnet-3.7, claude-sonnet-4 and so on.
4. Databricks Platform Integration (Display 1): The system utilizes Databricks Serving Endpoint to efficiently host and serve AI models, ensuring low-latency and high-concurrency inference responses to frontend requests. Additionally, it leverages Databricks App to build an interactive chatbot web interface, serving as the user-facing frontend for AI model interaction.



Display 1 Databricks Platform

Data Integration Layer

The Data Integration Layer shares a database with the Integrated SAS Programming Platform (ISPP) system, enabling real-time synchronization of macro programs, standard templates, specifications, study trackers, and other documents. Its functionalities include:

1. Real-Time Knowledge Mapping: Automatically suggests relevant macro programs and templates based on input, such as recommending the adsl_templete.sas when "adsl" is entered.
2. Standards Enforcement Engine: Automatically generates like program standard headers based on study tracker metadata.
3. Backend APIs: The backend is built using the Python Flask framework, utilizing APIs to proxy AI services from Databricks, and supports capabilities such as streaming output.

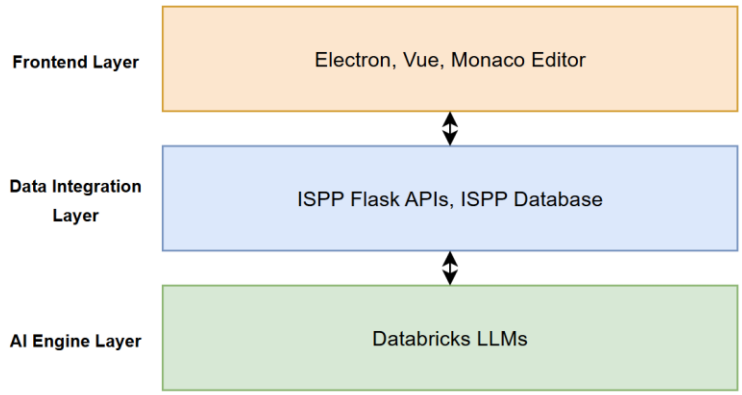


Figure 1 System Architecture in CodeGenius

TECHNICAL IMPLEMENTATION

During the development of CodeGenius, we encountered numerous technical challenges, such as adapting SAS syntax within the Monaco Editor, implementing AI streaming output, achieving auto-completion in the Monaco Editor, and enabling hover functionality for specific characters. In this article, we will delve into the solutions for implementing AI streaming output and hover functionality for specific characters.

AI streaming output

To address the AI streaming output, we leveraged the capabilities of the Python Flask framework to create a seamless connection between the frontend and the AI Engine Layer hosted on the Databricks platform. In frontend, Server-Sent Events (SSE) is used to stream real-time data from the server to the client. This approach allows the client to receive updates as soon as they are available, without the need for the client to repeatedly request new data.

First, initialize a streaming session and return the URL for streaming data. When receive a POST request from the front-end, extracting the request content and type. Then generate a unique session ID and store the request content in Redis. Construct a URL that the client can use to access the streaming data. Example code:

```
@bp.route('/stream_test/generate_code', methods=['POST'])
def get_stream_url():
    messages = request.json.get('content', '')
    session_id = str(uuid.uuid4())
    redis.rpush(session_id, messages)
    sse_url = f"ai/stream_test/generate_code_stream?session_id={session_id}"
    return jsonify({"sse_url": sse_url})
```

Once the frontend receives the URL for streaming data, an endpoint is established to handle streaming data transmission. This endpoint is responsible for receiving GET requests from the client and retrieving the session ID. It then fetches the request content from Redis and starts a new thread to call the Databricks API interaction function, processing the request in a non-blocking manner. A generator function is used to fetch data from Redis and stream it to the client.

A function is required to interact with the Databricks API and handle the streaming response. The HTTP request headers and body are constructed, a POST request is sent to the Databricks API with streaming enabled, and the response data is processed. Example Python code:

```
def databricks_api(messages, url, session_id):
    ......
    # Send the POST request with stream=True
    response = requests.post(url, data=data_json, headers=headers, stream=True, timeout=60)
    for line in response.iter_lines():
        if not line:
            continue
        if line.startswith('data:'):
            if line[5:].strip() == '[DONE]':
                redis.rpush(session_id, '__END__')
                return
            else:
                data = json.loads(line[5:])
                redis.rpush(session_id, f"data: {json.dumps(data)}\n\n")
redis.rpush(session_id,  '__END__')
```

Subsequently, the streaming data from the Databricks API is retrieved and transmitted to the client. Example Python code:

```
@bp.route('/stream_test/generate_code_stream', methods=['GET'])
def generate_code_stream():
    session_id = request.args.get('session_id')
    content = redis.lpop(session_id)
    ......
    def event_stream(session_id):
        while True:
            data = redis.lpop(session_id)
            if data is None:
                continue
            if data == b'__END__':
                break
            else:
                yield data
    thread = Thread(target=databricks_api, args=(messages, AI_API_URL, session_id))
    thread.start()
    return Response(event_stream(session_id), content_type='text/event-stream')
```

This setup ensures that the client receives real-time updates from the AI Engine Layer, providing a smooth and efficient streaming experience.

Hover for Specific Characters in Monaco Editor

To implement hover functionality for specific characters in Monaco Editor, particularly for displaying detailed descriptions and links when hovering over standard macros in CodeGenius, you can follow these steps:
First, define a Tokenizer for Specific Characters, You need to define a tokenizer that can identify the specific macros or characters in your SAS syntax. This will help in recognizing when a user hovers over these macros.

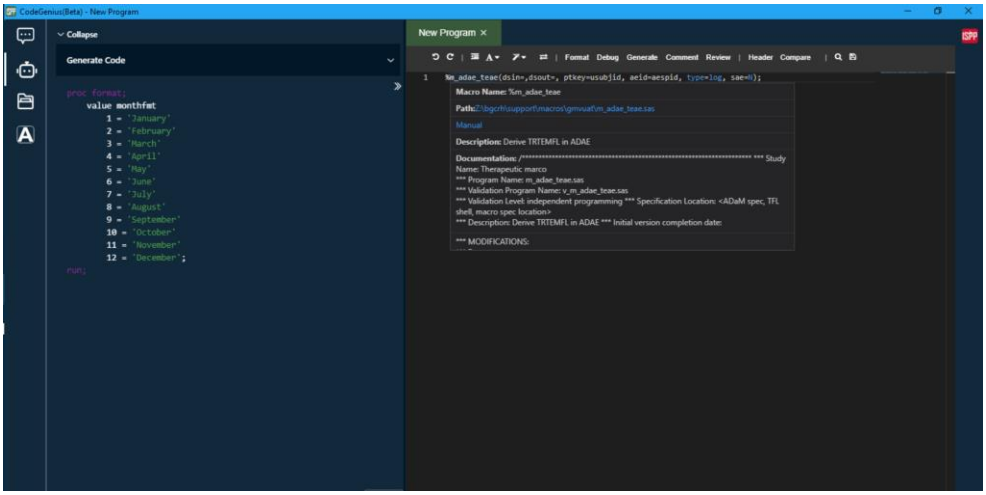
Second, register a Hover Provider, use monaco.languages.registerHoverProvider to register a hover event. This will allow you to specify what information should be displayed when the user hovers over the identified macros.

Last, provide Detailed Descriptions and Links, In the hover provider, return the content that includes a detailed description of the macro, a link to the user manual, and a link to open the macro file (Display 2).

Example JavaScript code:

```
monaco.languages.setMonarchTokensProvider('sas', {
    tokenizer: {
        root: [
            [/%\w+/, 'macro']
        ]
    }
});
// Register hover provider for SAS language
monaco.languages.registerHoverProvider('sas', {
    provideHover: function(model, position) {
        const word = model.getWordAtPosition(position);
        if (word && word.word.startsWith('%')) { // Check if it's a macro
            return {
                range: new monaco.Range(position.lineNumber, word.startColumn, position.lineNumber, word.endColumn),
                contents: [
                    { value: `**Macro: ${word.word}**` },
                    { value: 'This macro does XYZ. [User Manual](http://example.com/manual)' },
                    { value: '[Open Macro File](http://example.com/macrofile)' }
                ]
            };
        }
        return null;
    }
});
```

The final effect is as follows (Display 2).

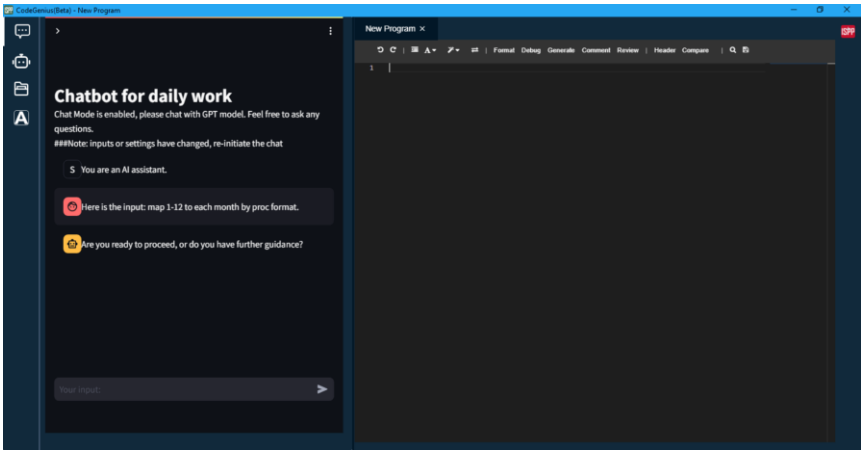


Display 2 Hover For Macro in CodeGenius

INNOVATIVE FEATURES AND IMPLEMENTATION

CUSTOMIZED EDITOR

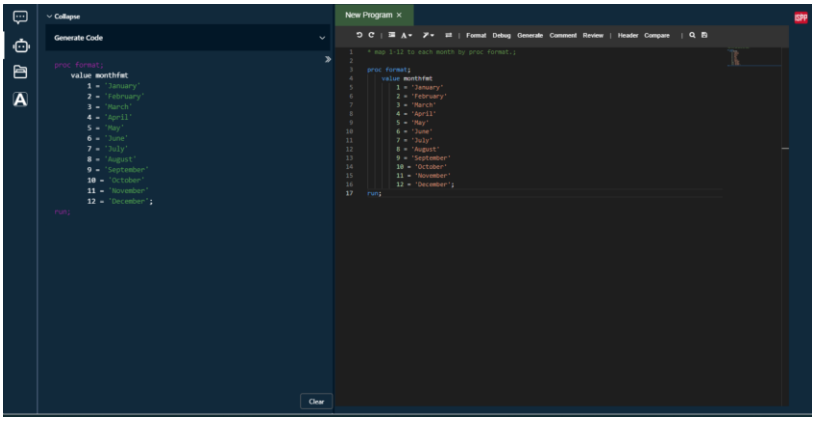
CodeGenius is built on the open-source Monaco Editor, offering a modern editing experience akin to Visual Studio Code. The user interface is intuitive and highly responsive, featuring syntax highlighting and code folding for the SAS language. Moreover, the editor's extensive customizability allows users to tailor settings to their personal preferences, such as fonts and themes, and supports features like text comparison highlighting. This adaptability caters to a wide range of complex programming needs, making the transition from the traditional SAS Enterprise Guide to CodeGenius much smoother.



Display 3 User Interface in CodeGenius

AI-DRIVEN CODE AUTOMATION

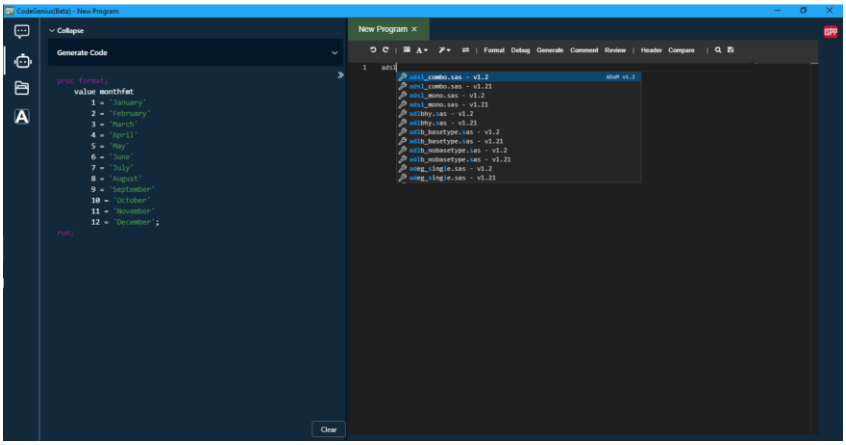
CodeGenius leverages advanced AI technology to automate various aspects of coding. Firstly, the system can automatically generate SAS code; users simply input basic instructions or descriptions, and the AI produces code snippets (Display 4). This not only accelerates the coding process but also reduces human errors. Additionally, CodeGenius offers code formatting features to ensure consistent style and enhance readability. The AI can also automatically add comments to the code, aiding users and team members in better understanding the code's logic. Lastly, the debugging feature intelligently analyzes the code to quickly identify and highlight potential errors, significantly improving code quality and stability.



Display 4 Generate Code by AI in CodeGenius

DOMAIN-SPECIFIC INTELLIGENT CODE COMPLETION

CodeGenius offers an intelligent code completion feature specifically designed for the field of statistical programming. By leveraging proprietary SAS macros and templates, the system provides highly customized code completion suggestions. For instance, when a user types "adsl," the system automatically retrieves and suggests the use of the company's ADSL ADaM template program (Display 5). This domain-specific completion capability not only enhances the efficiency of code development but also ensures accuracy and consistency across projects. By integrating these tailored suggestions, CodeGenius helps programmers adhere to best practices and company standards, ultimately streamlining the development process and reducing the likelihood of errors.



Display 5 Code Completion in CodeGenius

STANDARD AND KNOWLEDGE INTEGRATION FRAMEWORK

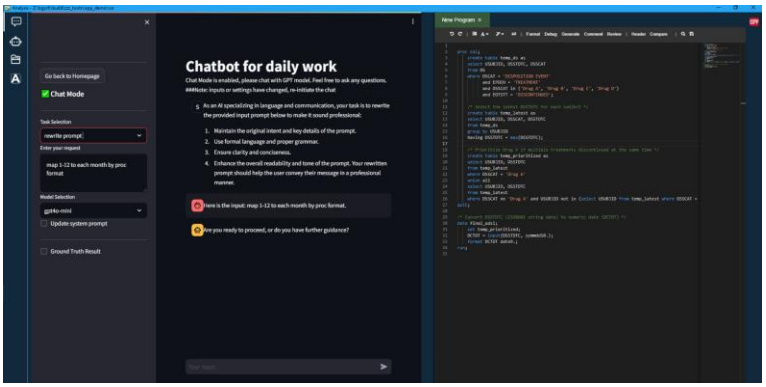
CodeGenius incorporates a robust framework that integrates company standards and knowledge repositories, ensuring that all code adheres to organizational guidelines. The system is equipped with built-in SAS program header templates and proprietary SAS macro programs, closely aligned with the department's ISPP platform, enabling users to swiftly apply company standards. Additionally, CodeGenius offers convenient hover tooltips that provide insights into the function of each macro and its parameters, along with direct links to the relevant user manuals. This feature also includes a one-click option to access company macros (Display 2), facilitating rapid learning and utilization of these resources.

This integration framework not only enhances code consistency but also fosters knowledge sharing and the swift implementation of standards across teams. By providing easy access to company-specific tools and documentation, CodeGenius empowers users to efficiently align their coding practices with organizational methodologies. This approach not only streamlines the development process but also

promotes collaboration and continuous learning within the organization, ensuring that best practices are consistently applied and that team members are well-equipped to leverage the full potential of the company's coding resources.

INTELLIGENT CHAT ASSISTANT AND CUSTOMIZABLE PROMPTS

CodeGenius features an intelligent chat assistant that supports free-form, multi-turn conversations, significantly enhancing the flexibility of user-system interactions. Users can communicate with the assistant using natural language to receive programming advice and resolve queries. Furthermore, the system allows users to customize prompts and switch between different LLM models to suit various programming needs and scenarios (Display 6). Behind the platform, it integrates the company’s macros and metadata related to SDTM, ADaM, and TFL, including various code functionalities, file paths, and other relevant information. This enables users to efficiently retrieve pertinent data and receive more effective suggestions and support. This personalized interaction not only improves the user experience but also transforms CodeGenius into a truly intelligent programming assistant.



Display 6 Chatbot in CodeGenius

FUTURE DEVELOPMENT

As an intelligent SAS editor, CodeGenius has already demonstrated its powerful capabilities and potential in the field of statistical programming. However, with the continuous advancement of technology and evolving user needs, CodeGenius is committed to ongoing innovation and improvement. In the future, we plan to further optimize the user experience by enhancing the system's responsiveness and stability.

In terms of AI capabilities, CodeGenius will continue to expand its functionalities to support a broader range of programming tasks. We plan to introduce more AI features, such as:

1. Leveraging the company's historical program library to accurately recommend and customize suitable reference programs for new studies.
2. Integrating AI Agent capabilities to automate the entire process from code review to debugging.
3. Exploring further automation to enable users to interact with the system using more natural language, allowing for direct automatic generation of corresponding programs and immediate presentation of results.

To enhance the practicality and adaptability of CodeGenius, we plan to strengthen its integration capabilities with other company systems. By seamlessly connecting with more internal and external platforms, CodeGenius will better support cross-platform dataflow and collaboration.

CONCLUSION

CodeGenius, as an intelligent SAS editor specifically designed for statistical programming, significantly enhances programming efficiency through its innovative features and intelligent design. Its AI-driven code automation capabilities allow for rapid generation, formatting, commenting, and debugging of code, reducing human errors and improving code quality. The domain-specific intelligent code completion

leverages internal SAS macros and templates to ensure code accuracy and consistency. Built on the Monaco Editor, CodeGenius offers a modern editing experience with a user interface similar to Visual Studio Code, enhancing the overall programming experience for users. The standard and knowledge integration framework ensures that code complies with organizational standards, promoting knowledge sharing across teams.

As an intelligent programming platform, CodeGenius is more than just an editor; it integrates powerful AI capabilities and internal company resources to provide a comprehensive programming solution. The value of CodeGenius lies not only in its robust features but also in its ability to adapt to the ever-changing technological landscape and evolving user needs, continuously providing support to its users.

CodeGenius is committed to closely monitoring advancements in AI technology, applying the latest technological explorations to practical business scenarios, and exploring further improvements in efficiency and quality within the SAS programming domain. By doing so, CodeGenius aims to remain at the forefront of intelligent programming tools, offering users a powerful and adaptable solution for their programming needs.

CONTACT INFORMATION

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

Yaohui Zhu
BeOne Medicines
yaohui.zhu@beigene.com
Xuejie Zhou
BeOne Medicines
xuejie.zhou@beigene.com