

A Framework for Compliant and Efficient LLM Integration in SAS Workflows

Enhancing Statistical Programming Efficiency in Pharma: Leveraging LLMs for Accelerated Development and Validation

Andy Ratcliffe | February 2026 | PHUSE SDE UK, Cambridge

Agenda

01

The Context

Understanding why the pharma industry requires enhanced LLM efficiency for statistical programming

02

LLM Fundamentals

Exploring what LLMs are and their critical role within GxP environments

03

Technical Bridge

Integrating LLMs securely with SAS via RAG and API access protocols

04

Accelerated Development

Practical LLM applications for code generation and comprehensive documentation

05

Compliant Validation

Leveraging LLMs for automated QC and independent verification scripts

06

Governance Framework

Risk mitigation strategies, trust establishment, and maintaining auditability

07

Future Directions

Summary of key takeaways and emerging opportunities in pharma programming

The Challenge: Speed, Quality, and Volume

The Pharma Data Treadmill

Increasing Complexity

Clinical trials are becoming more intricate, demanding sophisticated statistical methods and advanced programming logic to handle multifaceted data structures and regulatory requirements

Regulatory Bottlenecks

The imperative for 100% accuracy, strict GxP compliance, and manual validation processes significantly constrains turnaround times whilst maintaining the highest quality standards

The Programming Burden

Statistical programming tasks - creating SDTM, ADaM datasets, and TFLs - remain exceptionally high-effort phases with limited automation opportunities under current methodologies

The Strategic Goal

Maintain unwavering data integrity and regulatory compliance whilst drastically reducing development and validation cycle times through intelligent automation



LLMs: Your New Programming Assistant

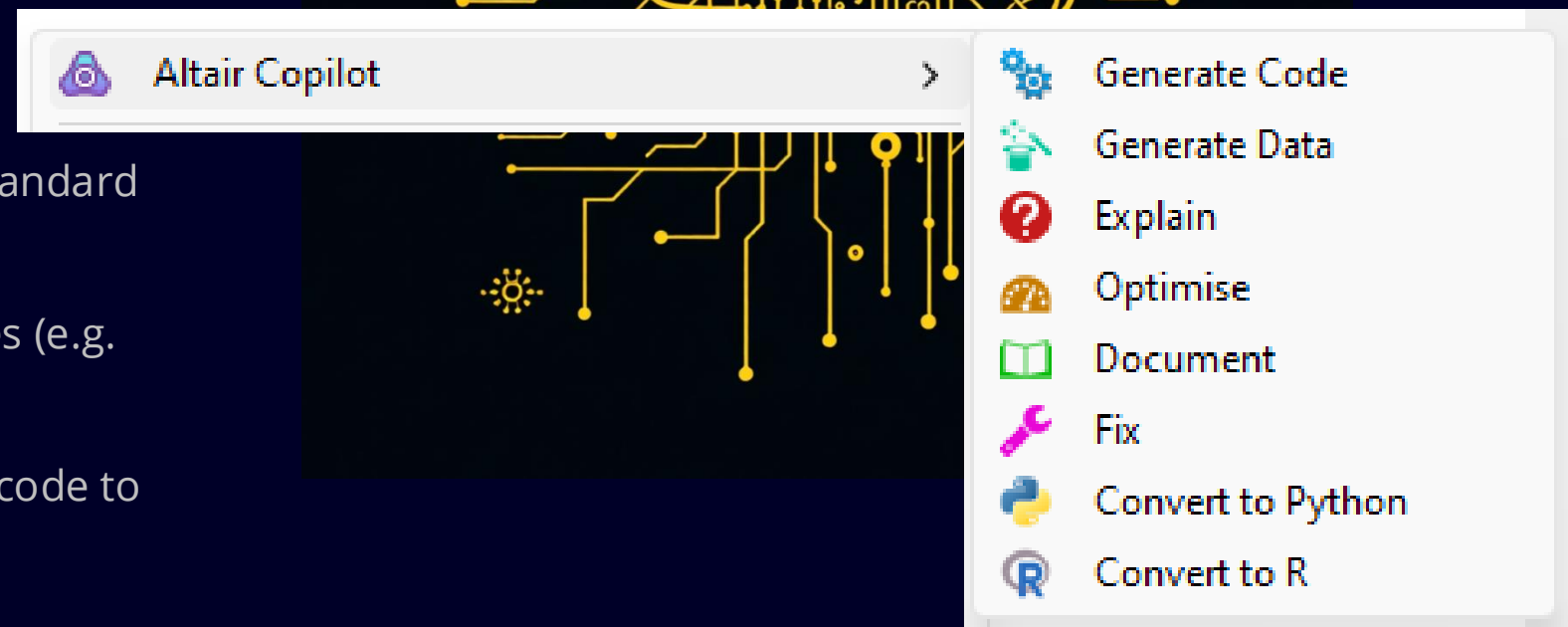
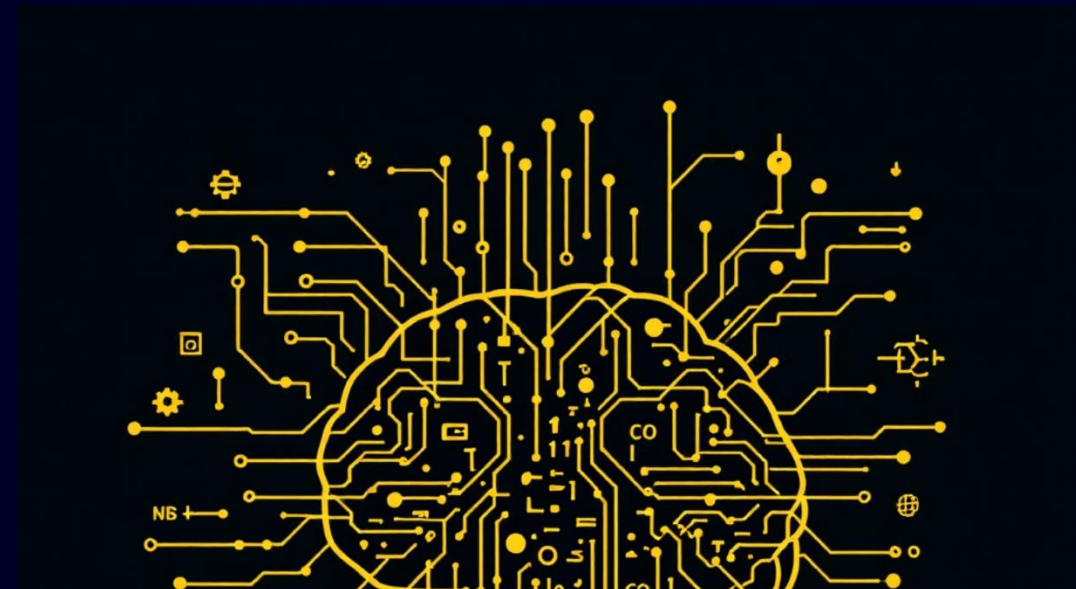
What is a Large Language Model?

Definition: Advanced algorithms such as ChatGPT, GPT-4, and Llama (built upon the Transformer architecture), representing the cutting edge of natural language processing technology.

Core Function: These models are trained on vast amounts of text to comprehensively understand, generate, and process both human language and programming code with remarkable fluency.

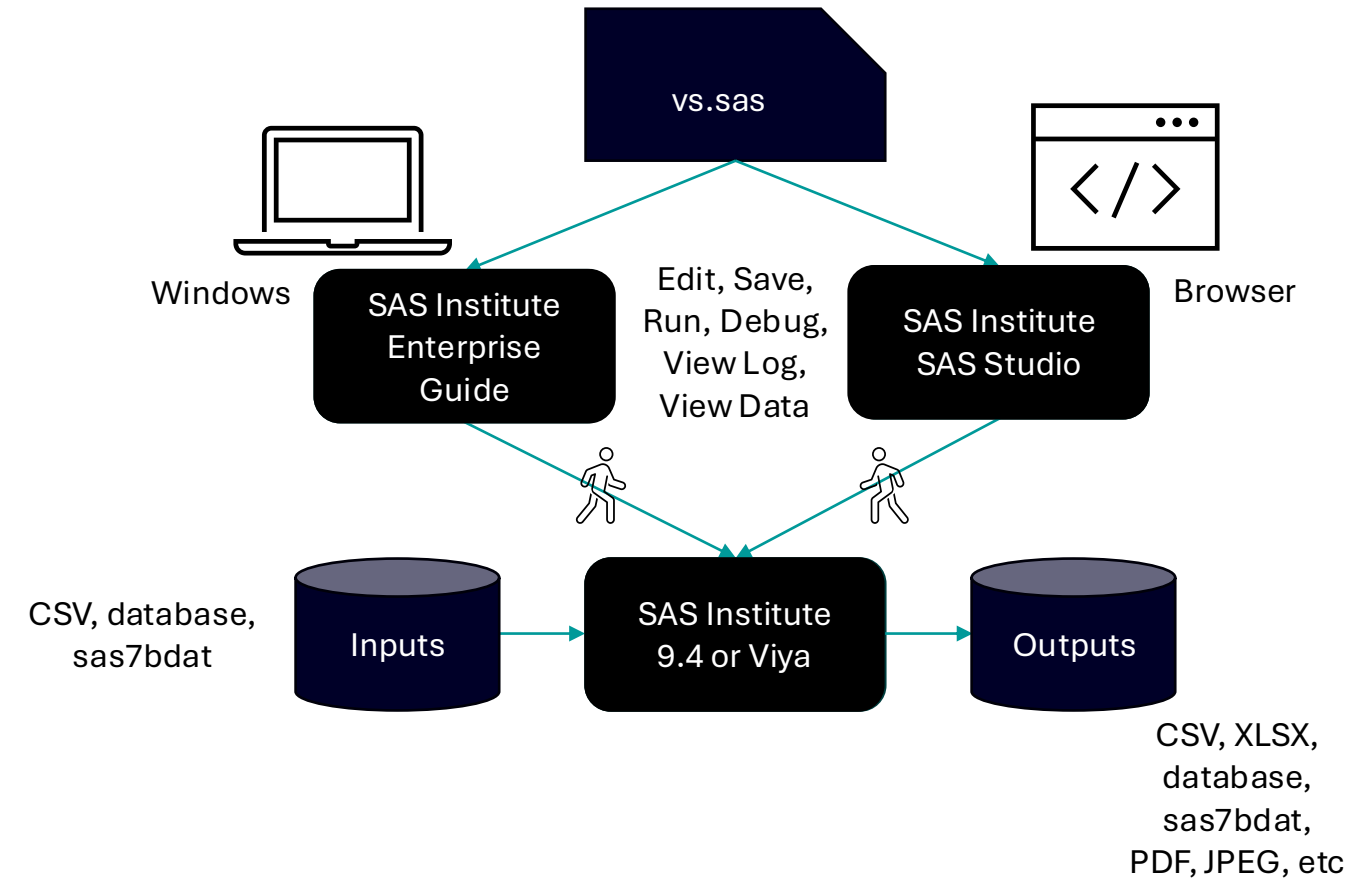
Key Capabilities in Programming:

- **Code Generation:** Automatically writing boilerplate code, standard functions, and procedure shells
- **Translation:** Converting code seamlessly between languages (e.g. SAS to R/Python)
- **Documentation:** Summarising or expanding upon existing code to enhance clarity and maintainability



SAS Language Compilers...

- Compilers read and execute the same SAS language code as your existing solution
- Altair provides equivalent software to your existing solution
 - Editors and compiler
- The compiler reads your program and performs the actions specified by your program



You have existing SAS language programs (files)...

You need to edit and run them, etc...

You can use EG on windows, or SAS Studio in your browser...

When you hit run, the editor passes your code to the SAS compiler to run it...

Running your code involves reading and writing data and results of various kinds...

Altair provides Workbench as your Windows-based editor – Mac too...

In your browser, Altair provides the Altair Web editor...

To run your code, Altair provides SLC, with the option of Altair's WPD data format...

Do I need to update all my code to work with SLC?? [No 😊]

LLMs: Your New Programming Assistant

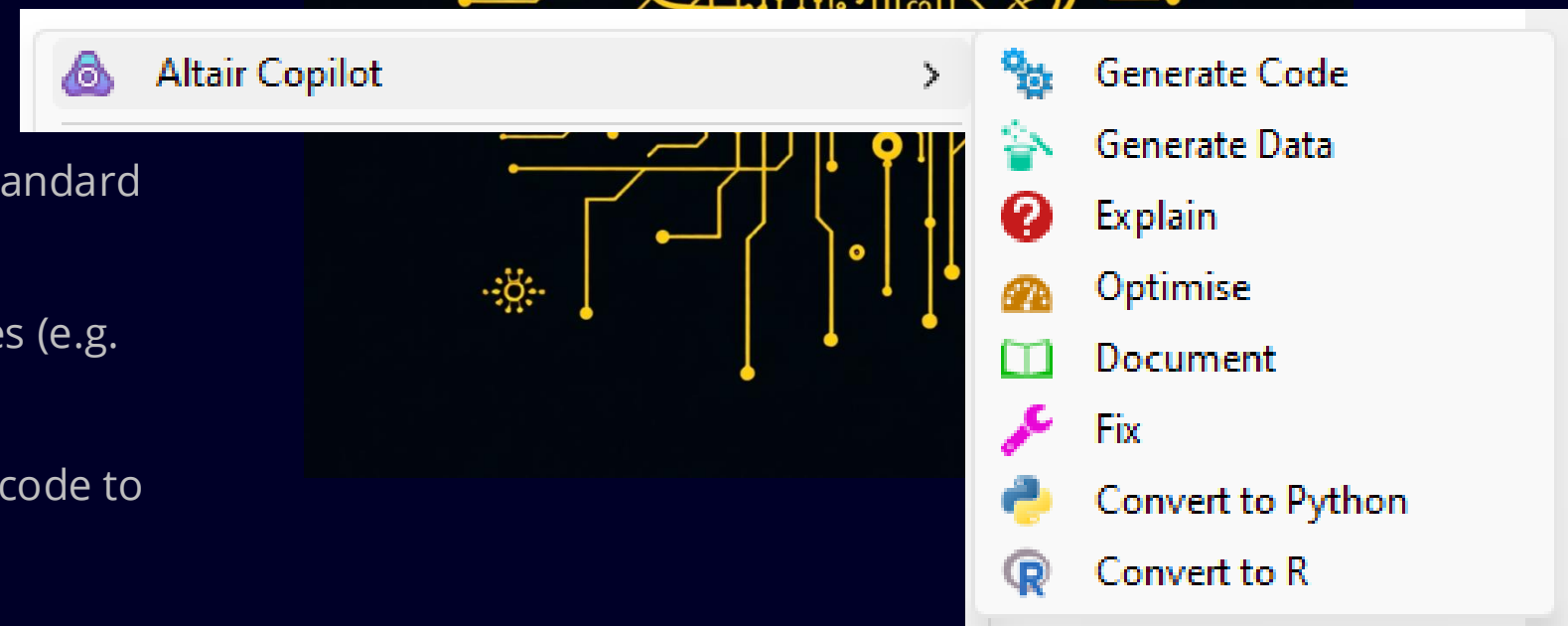
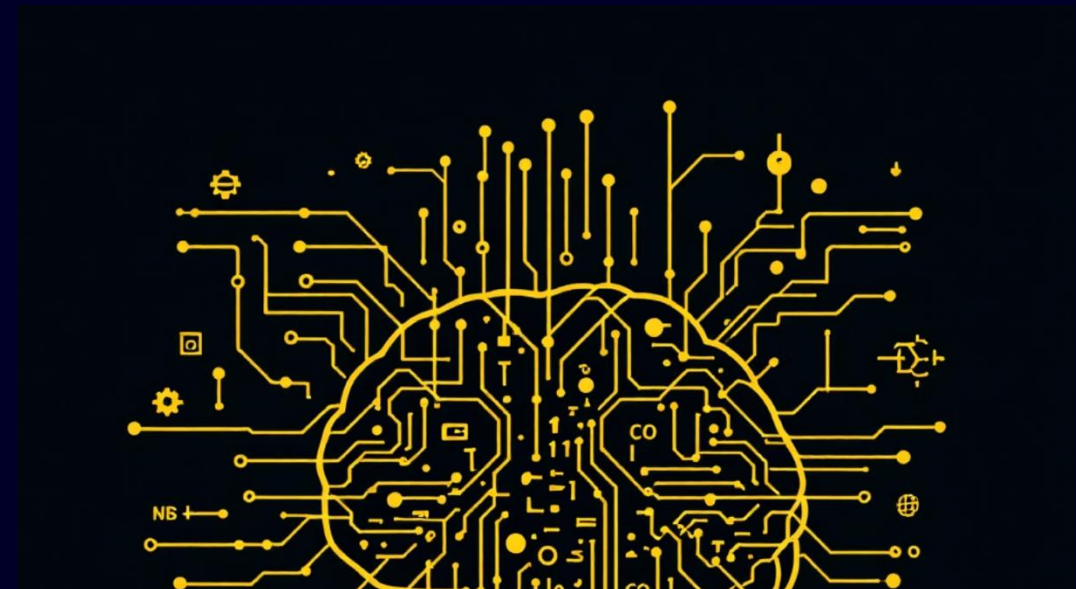
What is a Large Language Model?

Definition: Advanced algorithms such as ChatGPT, GPT-4, and Llama (built upon the Transformer architecture), representing the cutting edge of natural language processing technology.

Core Function: These models are trained on vast amounts of text to comprehensively understand, generate, and process both human language and programming code with remarkable fluency.

Key Capabilities in Programming:

- **Code Generation:** Automatically writing boilerplate code, standard functions, and procedure shells
- **Translation:** Converting code seamlessly between languages (e.g. SAS to R/Python)
- **Documentation:** Summarising or expanding upon existing code to enhance clarity and maintainability



LLMs: Your New Programming Assistant

The screenshot displays the Altair Analytics Workbench interface. At the top, a dark blue header contains the title "LLMs: Your New Programming Assistant". Below this, the main workspace shows a SAS script editor with a file explorer on the left and a log window on the right. The Altair Copilot sidebar is open on the left, showing a "Copilot" section with a "Ask Altair Copilot a question" input field. A yellow circle highlights a play button icon and a "0:00" timer. Below the input field, a status bar indicates "Hub: Not logged in".

Altair Copilot

- Generate Code
- Generate Data
- Explain
- Optimise
- Document
- Fix
- Convert to Python
- Convert to R

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

*Untitled 1.sas Rdemo1.sas SAS-R-Python... RUN_ALL_ALL.sas 02_DataPrep... adam_definex...

Local Server

- Databases
- Libraries
- Filerefs
- Log
- Hub Server
- nlb.altair-slc-dev.com2 Server

Altair Copilot

Copilot

Ask Altair Copilot a question

0:00

Altair Copilot uses AI. Results may not always be accurate.

Hub: Not logged in

Default Server: Local Server

Standard Licence

ENGLISH_UNITEDSTATES

No python interpreter

Progress Outputs

When Source File Results Server Status Durati...

12:03 23/10/2025

SIEMENS

LLMs: Your New Programming Assistant

Altair Copilot

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

File Explorer SLC Servers X Project Explorer Link Explorer Workflow Execution Order

Local Server
Databases
Libraries
File refs

Altair Copilot X

Copilot

Certainly! Below is the updated SAS program that reads the `sashelp.zipcode` dataset and determines the most frequent timezone.

```
/* Load the sashelp.zipcode dataset */
proc sql;
  create table zipcode as
  select *
  from sashelp.zipcode;
```

Ask Altair Copilot a question

Altair Copilot uses AI. Results may not always be accurate.

Hub: Not logged in Default Server: Local Server Writable Smart Insert 23 : 1 : 547 Standard Licence ENGLISH_UNITEDSTATES

Altair Copilot

*Untitled 1.sas X SAS-R-Python... »

```
1 /* Load the sashelp.zipcode dataset */
2 proc sql;
3   create table zipcode as
4   select *
5   from sashelp.zipcode;
6 quit;
7
8 /* Count the frequency of each timezone */
9 proc freq data=zipcode;
10   tables timezone / out=timezone_freq;
11 run;
12
13 /* Sort the frequency table to find the mo
14 proc sort data=timezone_freq;
15   by descending count;
16 run;
17
18 /* Print the most frequent timezone */
19 proc print data=timezone_freq(obs=1);
20   var timezone count;
21   title "Most Frequent Timezone in sashe
22 run;
23
```

Progress Outputs X

When Source File Re... Server S.. D.

No python interpreter

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

12:03

23/10/2025

SIEMENS

LLMs: Your New Programming Assistant

Altair Copilot

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

S × »4

Local Server

- Databases
- Libraries
- Filerefs
- Log
- Hub Server
- nlb.altair-slc-dev.co

*Untitled 1.sas × Rdemo1.sas SAS-R-Python.sas RUN_ALL_ALL.sas 02_DataPrep_2_2.... adam_definexml.... Macros.sas

1 /* @Copilot show me a table sorted by frequency of
2 timezone from sashelp.zipcode */
3

Log for Lc

NOTE: Cop
NOTE: Alt
Lic
NOTE: Thi

Altair...

Ask Altair Copilot a quest

Progress Outputs ×

When	Source	File	Results	Server	Status	Duration
		Writable	Smart Insert	3 : 1 : 87		
Hub: Not logged in			Default Server: Local Server	Standard Licence	ENGLISH_UNITEDSTATES	

No python interpreter

Windows Taskbar

12:04 23/10/2025

SIEMENS

LLMs: Your New Programming Assistant

Altair Copilot

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

Workspace1 - Generate Data - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

S × »4

Local Server

- Databases
- Libraries
- Filerefs
- Log
- Hub Server
- nlb.altair-slc-dev.co

*Untitled 1.sas Rdemo1.sas SAS-R-Python... RUN_ALL_ALL.sas 02_DataPrep... adam_definex... Generate Data X

Generate Data

Generate Data

Data

	Name	JobRole	Country
1	Bob Brown	Consultant	Netherlands
2	Alice Johnson	Nurse	Spain
3	John Doe	Consultant	Spain
4	Frank Moore	Administrator	Canada
5	Jane Smith	Technician	Germany
6	Grace Lee	Technician	Canada
7	Jane Smith	Researcher	UK
8	David Black	Technician	Netherlands
9	Eva Green	Technician	UK
10	John Doe	Lab Assistant	UK
11	Eva Green	Doctor	Netherlands
12	Henry Walker	Technician	UK
13	Carol White	Nurse	Spain
14	John Doe	Consultant	Spain
15	David Black	Nurse	Australia

Code

```
1 /* Generated by Altair Copilot. Check for errors. */
2 data output(keep=Name JobRole Country);
3   length Name $50 JobRole $50 Country $50;
4   array names[10] $50 _temporary_ ('John Doe',
5   array roles[10] $50 _temporary_ ('Doctor', 'N
6   array countries[10] $50 _temporary_ ('USA', '
7
8   do i = 1 to 50;
9     Name = names[ceil(ranuni(12345)*10)];
10    JobRole = roles[ceil(ranuni(12345)*10)];
11    Country = countries[ceil(ranuni(12345)*10)];
12    output;
13  end;
14 run;
15 /* End of generated code. */
```

Log for Lo

NOTE: Cop

NOTE: Alt

NOTE: Lic

NOTE: Thi

Altair...

Ask Altair Copilot a quest

Progress Outputs X

When	Source	File	Results	Server	Status	Duration
------	--------	------	---------	--------	--------	----------

Hub: Not logged in Default Server: Local Server Standard Licence ENGLISH_UNITEDSTATES

No python interpreter

12:04

23/10/2025

SIEMENS

LLMs: Your New Programming Assistant

Altair Copilot

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

File Explorer SLC Servers X Project Explorer Link Explorer Workflow Execution Or...

Local Server

Databases

Libraries

Filerefs

Altair Copilot X

PROC SORT procedure to sort the dataset `timezone_freq`.

- `by descending count;` : This specifies that the dataset should be sorted by the `count` variable in descending order. The `count` variable represents the frequency of each `timezone`.
- `run;` : This statement executes the procedure.

4. Print Procedure:

```
proc print data=timezone_freq;
run;
```

Ask Altair Copilot a question

Altair Copilot uses AI. Results may not always be accurate.

*Untitled 1.sas X Rdemo1.sas SAS-R-Python... »4

```
1 /* show me a table sorted by frequency of
2   timezone from sashelp.zipcode */
3 /* Generated by Altair Copilot. Check for errors. */
4 proc freq data=sashelp.zipcode;
5     tables timezone / out=timezone_freq;
6 run;
7
8 proc sort data=timezone_freq;
9     by descending count;
10 run;
11
12 proc print data=timezone_freq;
13 run;
14 /* End of generated code. */
15
```

Progress Outputs X

When Source File Res... Server St... D...

No python interpreter

Hub: Not logged in Default Server: Local Server Standard Licence ENGLISH_UNITEDSTATES

12:05 23/10/2025

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

SIEMENS

LLMs: Your New Programming Assistant

Altair Copilot

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

File Explorer SLC Servers X Project Explorer Link Explorer Workflow Execution Or...

Local Server

Databases

Libraries

Filerefs

Altair Copilot X

Copilot

To optimize the given SAS code, we can combine the PROC FREQ and PROC SORT steps into a single step using the ORDER=FREQ option in PROC FREQ to sort the results by frequency directly. This eliminates the need for a separate sorting step. Here's the optimized code:

```
/* Show me a table sorted by frequency of timezone from sashelp.zipcode */  
proc freq data=sashelp.zipcode order=freq;  
    tables timezone / out=timezone_freq;
```

Ask Altair Copilot a question

Altair Copilot uses AI. Results may not always be accurate.

*Untitled 1.sas X Rdemo1.sas SAS-R-Python... »4

```
1 /* show me a table sorted by frequency of  
2   timezone from sashelp.zipcode */  
3 /* Generated by Altair Copilot. Check for errors. */  
4 proc freq data=sashelp.zipcode;  
5     tables timezone / out=timezone_freq;  
6 run;  
7  
8 proc sort data=timezone_freq;  
9     by descending count;  
10 run;  
11  
12 proc print data=timezone_freq;  
13 run;  
14 /* End of generated code. */  
15
```

Progress Outputs X

When Source File Res... Server St... D...

No python interpreter

Hub: Not logged in Default Server: Local Server Standard Licence ENGLISH_UNITEDSTATES

12:05
23/10/2025

SIEMENS

LLMs: Your New Programming Assistant

Workspace1 - andy1/Rdemo1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

Local Server

- Databases
- Libraries
- Filerefs
- Log
- Hub Server
- nlb.altair-slc-dev.com2 Server

Altair Copilot

Copilot

```
4
5 /* Start a PROC R block to execute R code within SAS */
6 proc r;
7     /* Submit R code to be executed */
8     submit;
9         /* Print "Hello world" in R */
10        print("Hello world")
11    /* End the submission of R code */
12    endsubmit;
13 run;
14
15 /* Start another PROC R block to execute more R code within SAS */
16 PROC R;
17     /* Submit R code to be executed */
18     SUBMIT;
19         /* Load the datasets library in R */
20         library(datasets)
21         /* Load the iris dataset in R */
22         data(iris)
23         /* Generate a summary of the iris dataset in R */
24         summary(iris)
25         /* Plot the iris dataset in R */
26         plot(iris)
27     /* End the submission of R code */
28     ENDSUBMIT;
29 RUN;
```

Progress Outputs

When	Source	File	Results	Server	Status	Durati...
------	--------	------	---------	--------	--------	-----------

Altair Copilot uses AI. Results may not be accurate.

Hub: Not logged in

Default Server: Local Server

Standard Licence

ENGLISH_UNITEDSTATES

No python interpreter

12:05
23/10/2025

- Generate Code
- Generate Data
- Explain
- Optimise
- Document
- Fix
- Convert to Python
- Convert to R

LLMs: Your New Programming Assistant

Altair Copilot

Workspace1 - andy1/SAS-R-Python.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

File Explorer SLC Servers X Project Explorer Link Explorer Workflow Execution Order

Local Server

- Databases
- Libraries

Altair Copilot X

You

▶ Suggest fixes for the following SAS language code

Copilot

Certainly! Here are the suggested fixes for the provided SAS code:

- Fixing the `proc print` statement:
 - The `proc print` statement does not have a `sort` option. If you want to sort the data before printing, you should use the `proc sort` statement first.
- Fixing the commented options statement:

Ask Altair Copilot a question

Altair Copilot uses AI. Results may not always be accurate.

Progress Outputs X

When Source File Res... Server S... D...

2 : 32 [14]

No python interpreter

Hub: Not logged in Default Server: Local Server Standard Licence ENGLISH_UNITEDSTATES

```
1
2 proc print data=sashelp.vlibnam sort=libname;
3 run;
4
5 *options set = pythonpath "c:\users\aratchcliffe\appda
6 proc python;
7 submit;
8 import numpy as np
9 print(np.__version__)
10 import pandas as pd
11 print(pd.__version__)
12 print ("Hello world")
13 endsubmit;
14 run;
```

- Generate Code
- Generate Data
- Explain
- Optimise
- Document
- Fix
- Convert to Python
- Convert to R

LLMs: Your New Programming Assistant

Altair Copilot

Sort by frequency in descending order
timezone_freq = timezone_freq.sort_values(by='count',
ascending=False)

Print the sorted frequency table
print(timezone_freq)

Make sure to replace 'sashelp_zipcode.csv' with the actual path to your
dataset file. This code reads the dataset, calculates the frequency of each timezone,
sorts the frequencies in descending order, and prints the result.

Ask Altair Copilot a question

Altair Copilot uses AI. Results may not always be accurate.

Workspace1 - Untitled 1.sas - Altair Analytics Workbench

File Edit Navigate Search Project Run Altair Altair Copilot Window Help

File Explorer SLC Servers Project Explorer Link Explorer Workflow Execution Order

Local Server
Databases
Libraries

Altair Copilot

*Untitled 1.sas *Rdemo1.sas SAS-R-Python...

1 /* show me a table sorted by frequency of
2 timezone from sashelp.zipcode */
3 /* Generated by Altair Copilot. Check for errors. */
4 proc freq data=sashelp.zipcode;
5 tables timezone / out=timezone_freq;
6 run;
7
8 proc sort data=timezone_freq;
9 by descending count;
10 run;
11
12 proc print data=timezone_freq;
13 run;
14 /* End of generated code. */
15

Progress Outputs

When Source File Res... Server S... D...

No python interpreter

Generate Code

Generate Data

Explain

Optimise

Document

Fix

Convert to Python

Convert to R

Hub: Not logged in

Default Server: Local Server

Standard Licence

ENGLISH_UNITEDSTATES

12:06
23/10/2025

SIEMENS



LLMs in a High-Stakes Environment

The GxP Distinction: Stochastic vs. Deterministic

Stochastic Nature

LLMs are fundamentally **probabilistic**. They predict the "most likely" correct answer through statistical patterns, which can occasionally lead to "**hallucinations**", i.e. outputs that appear plausible but are factually incorrect or inconsistent

Deterministic Requirement

Regulated programming under GxP mandates **deterministic** results—outputs must be perfectly consistent, reproducible, and verifiable across all executions without variation

The Human-in-the-Loop (HIL) Model

1

LLMs as Tools

Productivity assistants that augment human capability

2

Programmers as SMEs

Statistical programmers retain sole ownership of validation and compliance

3

Mandatory Review

All LLM-generated code requires rigorous review, testing, and validation

The Technical Bridge: Connecting LLMs to Your Workflow

Integrating LLMs with SAS Infrastructure

External API Access

LLMs reside outside the SAS environment, requiring secure API communication channels to facilitate interaction whilst maintaining system integrity

- **SAS language:** Utilising PROC HTTP or PROC PYTHON to send structured prompts to secure LLM endpoints (e.g. AWS Bedrock, Azure OpenAI Service)
 - “Bring Your Own LLM”
 - Governance and Explainability
 - Low-Code / No-Code Tools
- Encrypted connections ensure data protection during transmission
- Authentication protocols prevent unauthorised access



❏ Data Security is Paramount

- **NEVER** send clinical patient data to the LLM under any circumstances
- LLMs only process **metadata** (variable lists, formats), **code snippets**, or **specifications**
- Implementation must use **private LLM instances** (e.g. within a private cloud/endpoint) to maintain environment security and regulatory compliance

Retrieval-Augmented Generation (RAG) for Pharma Standards

RAG: The Compliance Guardrail

The RAG Concept

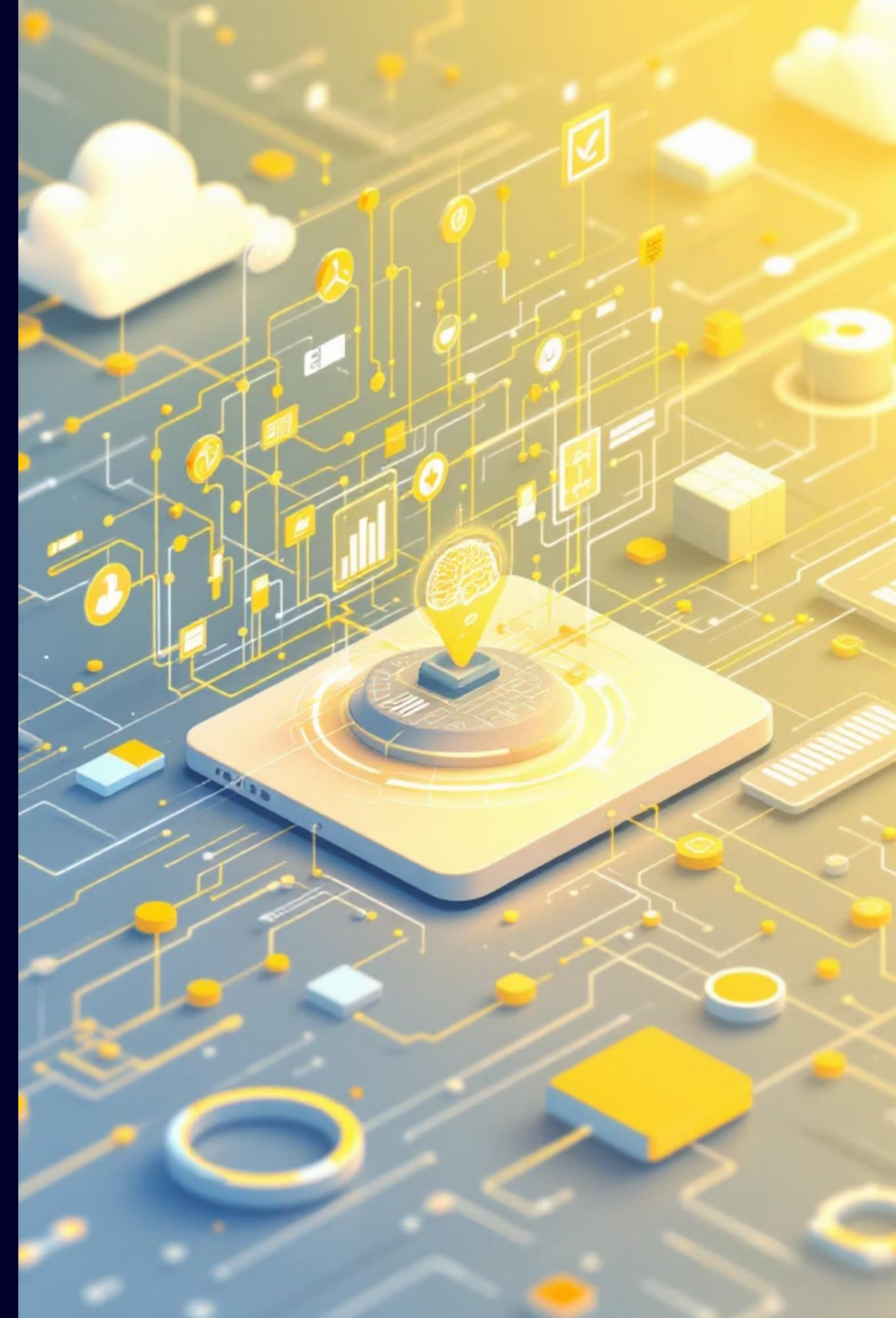
Whilst LLMs are exceptionally powerful, they remain generic without context. RAG grounds the LLM with specific, authoritative knowledge tailored to your organisational requirements.

This approach transforms a general-purpose AI into a **domain-specific programming expert** that understands your unique standards and compliance frameworks.

The Benefit: By referencing internal standards, the LLM generates code that is not merely syntactically correct, but also **institutionally compliant** and audit-ready from the outset.

Pharma RAG Knowledge Base

- Internal Standard Operating Procedures (SOPs) defining organisational best practices
- Internal Programming Guidelines and comprehensive Style Guides
- Validated SAS Macros and Utility Functions from your approved library
- CDISC Implementation Guides (SDTM/ADaM/Define-XML) ensuring industry standards



Use Case I: Accelerating Development & Reducing Boilerplate

LLMs in Accelerated Development (I) - Code Generation

40-60%

Time Reduction

Potential decrease in time spent on initial drafting of routine SAS modules

3x

Productivity Gain

Faster iteration cycles for standard programming tasks

Challenge Addressed: Writing Repetitive, Low-Variability Code

1

TFL Shells

Generating empty PROC REPORT or PROC TABULATE frameworks directly from Display Mock specifications, eliminating hours of manual template creation.

2

ADaM Logic

Creating DATA steps for standard derivations (e.g. VISIT Day/Week calculations based on protocol logic), ensuring consistency across studies.

3

Variable Formatting

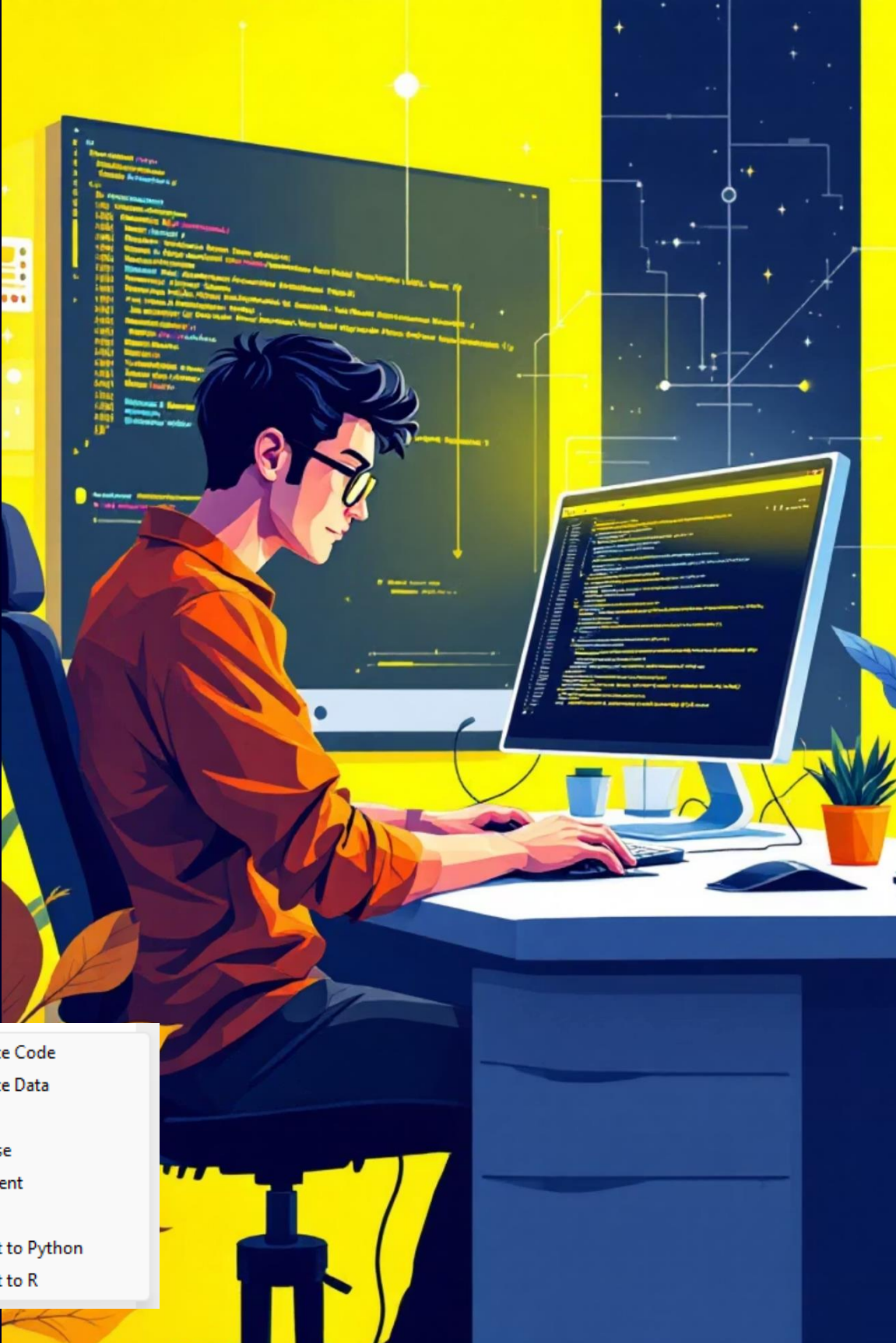
Generating comprehensive PROC FORMAT code based on value-level specifications, reducing formatting errors and improving documentation quality.



Altair Copilot



- Generate Code
- Generate Data
- Explain
- Optimise
- Document
- Fix
- Convert to Python
- Convert to R



Use Case II: Debugging, Translation, and Documentation

LLMs in Accelerated Development (II) - Beyond Code Generation



Debugging & Error Resolution

Feed a SAS log error message *and* the relevant code snippet to the LLM. The model suggests targeted fixes, citing potential causes based on its extensive training data or RAG-fed SOPs, dramatically reducing troubleshooting time.



Code Translation & Migration

Quickly convert legacy SAS procedures to modern open-source equivalents (R/Python) for hybrid systems. Translate complex SQL logic into SAS Data Steps for consistent data flow across platforms.



Documentation Generation

Automatically draft GxP-compliant programming specification documents from high-level analysis plan text, ensuring consistency between analysis intent and code implementation whilst reducing documentation burden.



LLMs in Compliant Validation (I)

Use Case III: Validation & Quality Control (QC)

The Validation Goal: Validation shifts from purely checking execution to ensuring *compliance with standards*.

LLM-Assisted Automated QC

- The LLM acts as a "smart" reviewer by ingesting the human-written code
- It checks the code against the RAG-fed **Internal Style Guide and SOPs**
- **Output:** A checklist flagging inconsistencies (e.g. missing header comments, non-standard variable names, non-compliant macro usage)

Benefit: Allows human QC effort to focus on statistical logic and scientific interpretation, not stylistic errors.





LLMs in Compliant Validation (II)

Use Case IV: Creating Independent Verification Scripts

The Golden Rule of Validation: Verification code must be written *independently* from the development code

1

Prompt 1 (Development)

Programmer writes code with human-in-the-loop oversight, ensuring logic aligns with analysis specifications

2

Prompt 2 (Validation)

QA/Reviewer uses the *original analysis specification* to prompt the LLM to generate a **separate, single-purpose R or SAS script**

3

Verification Task

This script reproduces a key output (e.g. calculating N or Mean of an endpoint) using an alternative method or language

4

Audit Trail

Generate comparison documentation (Diff Reports) between the main programme's output and the LLM verification script's output



Risk Mitigation and Governance

Governing LLM Use: Trust, Traceability, and Training

1

Hallucination Management

Implement rigorous testing to proactively identify and block illogical outputs (e.g. testing with negative/illogical prompts).

2

Audit Trail & Traceability

All LLM interactions must be logged, including the exact prompt, the model used, and the raw output, to support regulatory audits.

3

Training & Prompt Literacy

Programmers and QA must be trained on **Prompt Engineering** - how to write clear, constrained, and effective prompts that minimise ambiguity and risk.

4

Continuous Monitoring

Establish metrics to continuously monitor the accuracy and compliance rate of the LLM in production environments.

Conclusion

Summary: The New Paradigm

Component	Traditional Workflow	LLM-Augmented Workflow
Development	Manual coding, low automation	Accelerated drafting, boilerplate reduction
Code Quality	Dependent on manual review	Automated QC against RAG-fed SOPs
Validation	100% human-driven dual programming	LLM assists in generating independent verification scripts
Speed	Linear scale with complexity	Exponentially reduced cycle times

Key Takeaway

LLMs are powerful accelerators, but success is contingent on secure integration, robust compliance guardrails (RAG), and a commitment to the Human-in-the-Loop model.

Future Directions & Q&A

The Next Steps

- 1 **Deeper Integration**
Moving LLM assistance directly into IDEs (Integrated Development Environments) for real-time coding suggestions
- 2 **CDISC Automation**
Full LLM-driven generation of Define-XML and annotated CRF (aCRF) from protocols
- 3 **Continuous Accuracy Monitoring**
Developing sophisticated systems to track and retrain domain-specific models

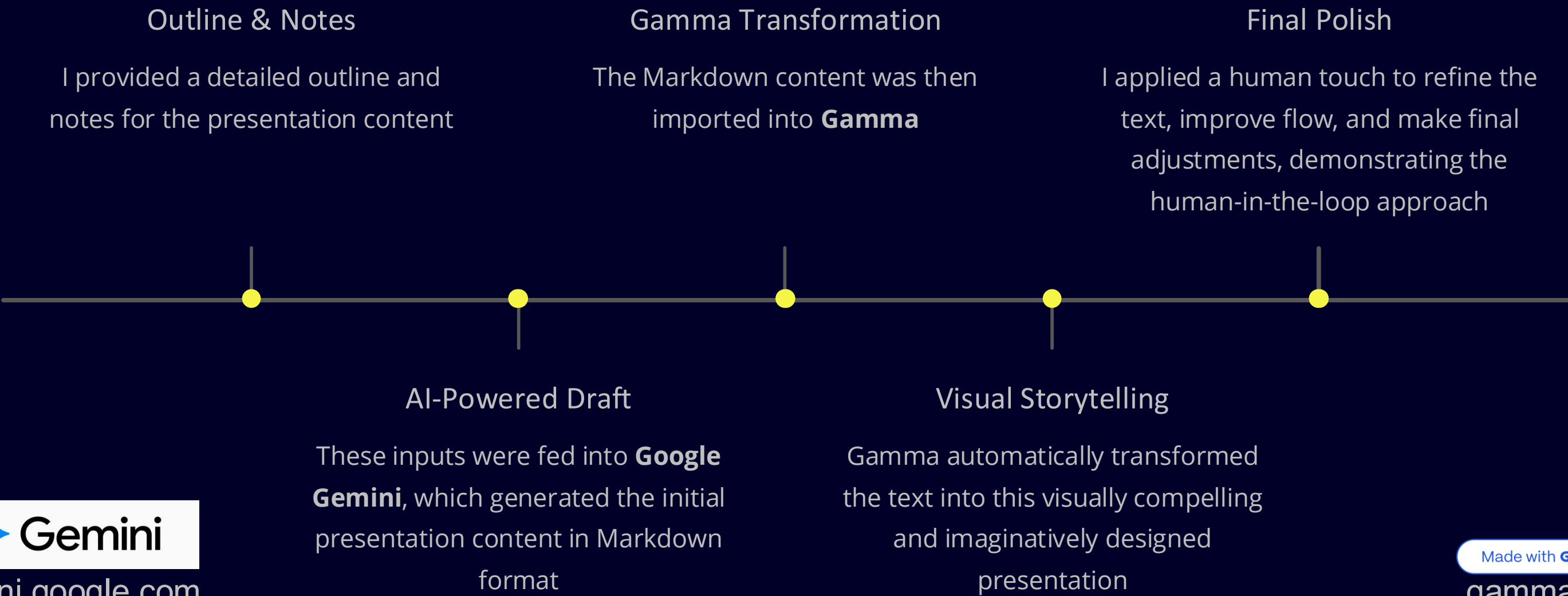


The Future: AI will not replace the statistical programmer, but the programmer who uses AI will replace the one who doesn't

Questions & Discussion

Appendix: How This Presentation Was Created

This presentation itself is a demonstration of how AI can enhance efficiency and creativity in content generation...



gemini.google.com

Made with **GAMMA**

gamma.app

SIEMENS

Thank You



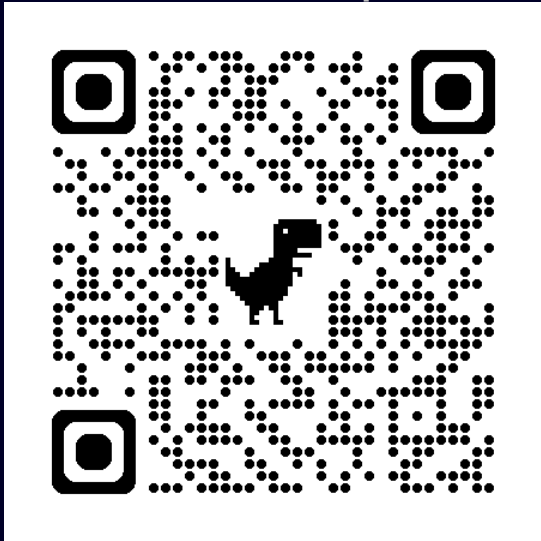
Andy Ratcliffe
PreSales Solutions Consultant
Siemens Industry Software Inc.
Andy.Ratcliffe@siemens.com
<https://www.linkedin.com/in/andrewratcliffeuk/>



Andrew Ratcliffe
Passionate, delivery-focused
life sciences leader, across techn...



MeetUp



Independent SAS Language Community
<https://www.linkedin.com/company/suguki/>

Networking events for the SAS Language Community. We hold meetups for anyone interested in SAS - be that programming, administration, analytics, modelling or just for fun.
<https://www.meetup.com/SUGUKI/>

LinkedIn



