PHUSE SDE Conference • Cambridge • 2026

Building Practical Tools in the LLM Era

From Idea to Functioning Solution in Minutes

Dr Damian Wierzbicki
Statistical Programmer, Phastar
damian.wierzbicki@phastar.com

phastar

The Situation We All Face

> “A task begs for automation, yet building a tool traditionally requires deep programming expertise and more time than the task itself warrants.”

Hours

To build something “simple”

Expertise

Required in languages outside our domain

ROI?

Often not worth the investment

Large language models have fundamentally changed this equation.

phastar

The Categories of AI-Assisted Tools

CATEGORY 1

API-Based Tools

Technical tools requiring precision, integration with existing systems, and programmatic control.

Example

SAS Macro Helper

✓ Multi-model support
✓ Customizable prompts
✓ Integrated into workflow

CATEGORY 2

Prompt-Based Tools

General-purpose productivity tools built through iterative conversation with an LLM.

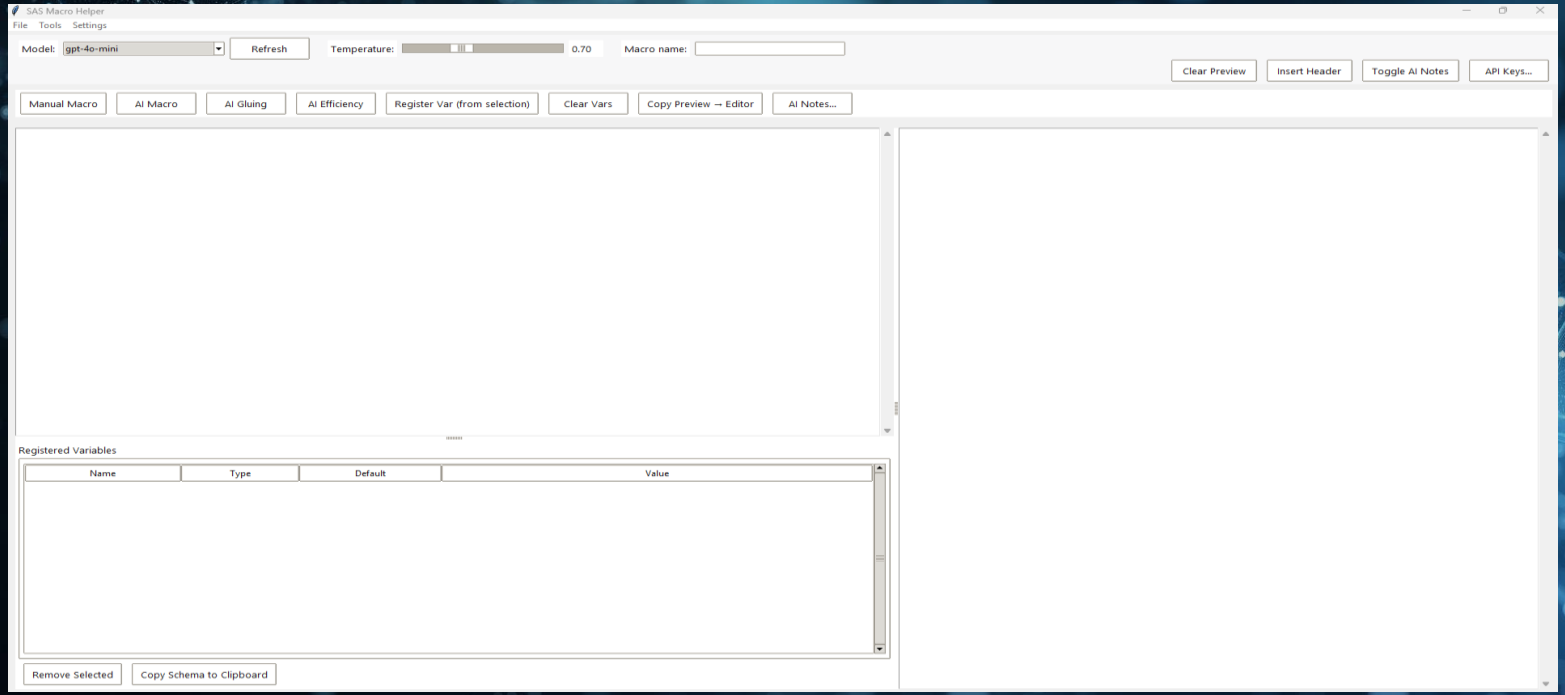
Example

Document Organizer

✓ Built in a single session
✓ Iterative refinement
✓ No API knowledge needed

Category 1: SAS Macro Helper

A desktop application that leverages LLM APIs to generate, optimize, and consolidate SAS macros for clinical trials programming.



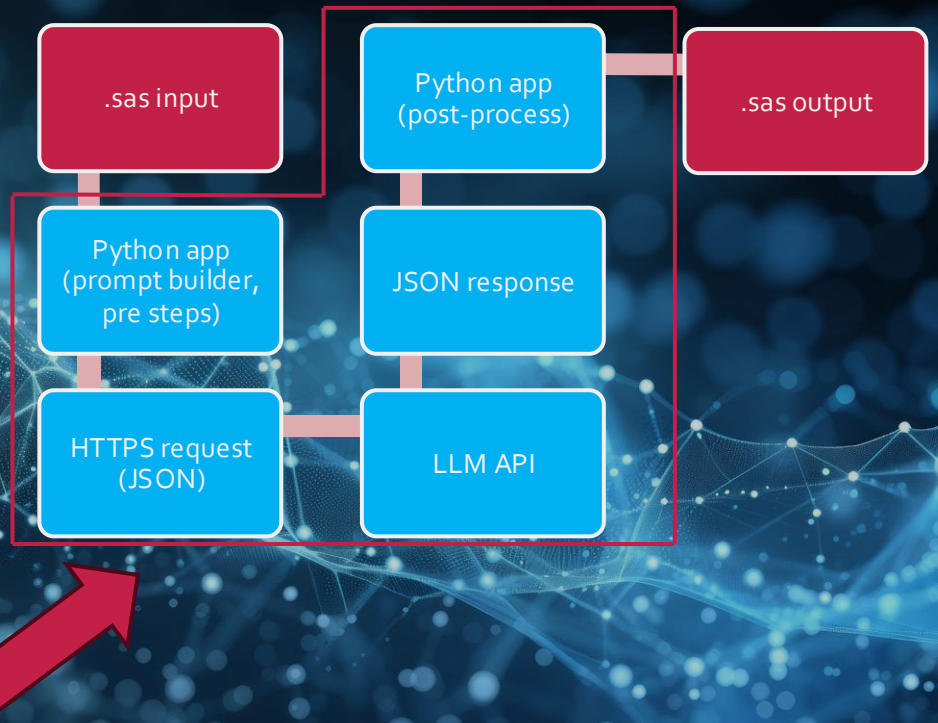
Programmatic, not chat: how the API works

What's an API (for LLMs)?

Your code sends an HTTPS **request** (JSON) and gets a **response** (JSON text/code).

Auth via API key; you pick the **model** and **parameters.**

Official Python SDK wraps the HTTP calls for you.



phastar

Why API, Not Just a Chatbot?

CHAT UI LIMITATIONS

Typing fatigue — retyping prompts, managing windows, copy-paste errors

Prompt drift — tiny wording changes produce different outputs

Limited control — hidden defaults, restricted model/temperature settings

No automation — can't add pre/post-processing steps

PROGRAMMATIC API

Repeatability — same prompt template, consistent results

Full control — choose model, temperature, system prompt

Pre/Post steps — parse output, strip markdown, add headers

Multi-provider — OpenAI, Claude, Gemini from one interface

phastar

The Reality of SAS Programming

We write **near-duplicate programs/routines** constantly: same logic, slightly different parameters, datasets, or variables.

Manual macro creation is **time-consuming**: identifying parameters, refactoring code, adding defaults and checks.

Code reviews often reveal **efficiency opportunities** we didn't have time to implement.

THE QUESTION

Can LLMs help us **automate the tedious parts** while we stay in control?

phastar

SAS Macro Helper — Three Modes

01

Convert to Macro

Register variables manually, then let AI wrap your working code into a reusable %macro with defaults and checks.

```
input → parameters → %macro
```

02

Glue Programs

Load 2-3 similar programs (e.g., lab chem vs. hematology). AI generalizes them into one macro.

```
prog1 + prog2 → unified %macro
```

03

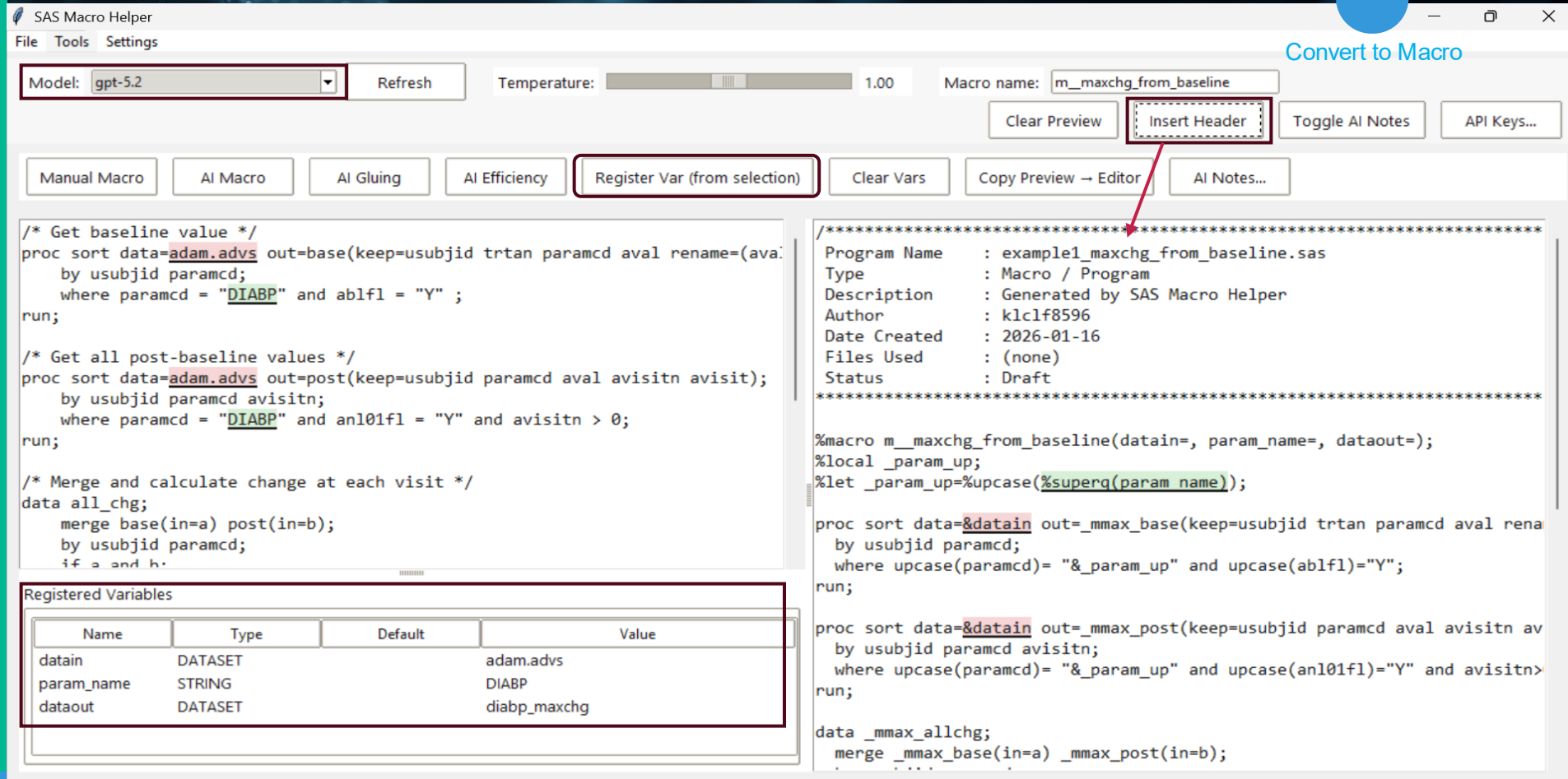
Efficiency Review

AI suggests safe cleanups: arrays, loops, deduplication. Behavior preserved, structure improved.

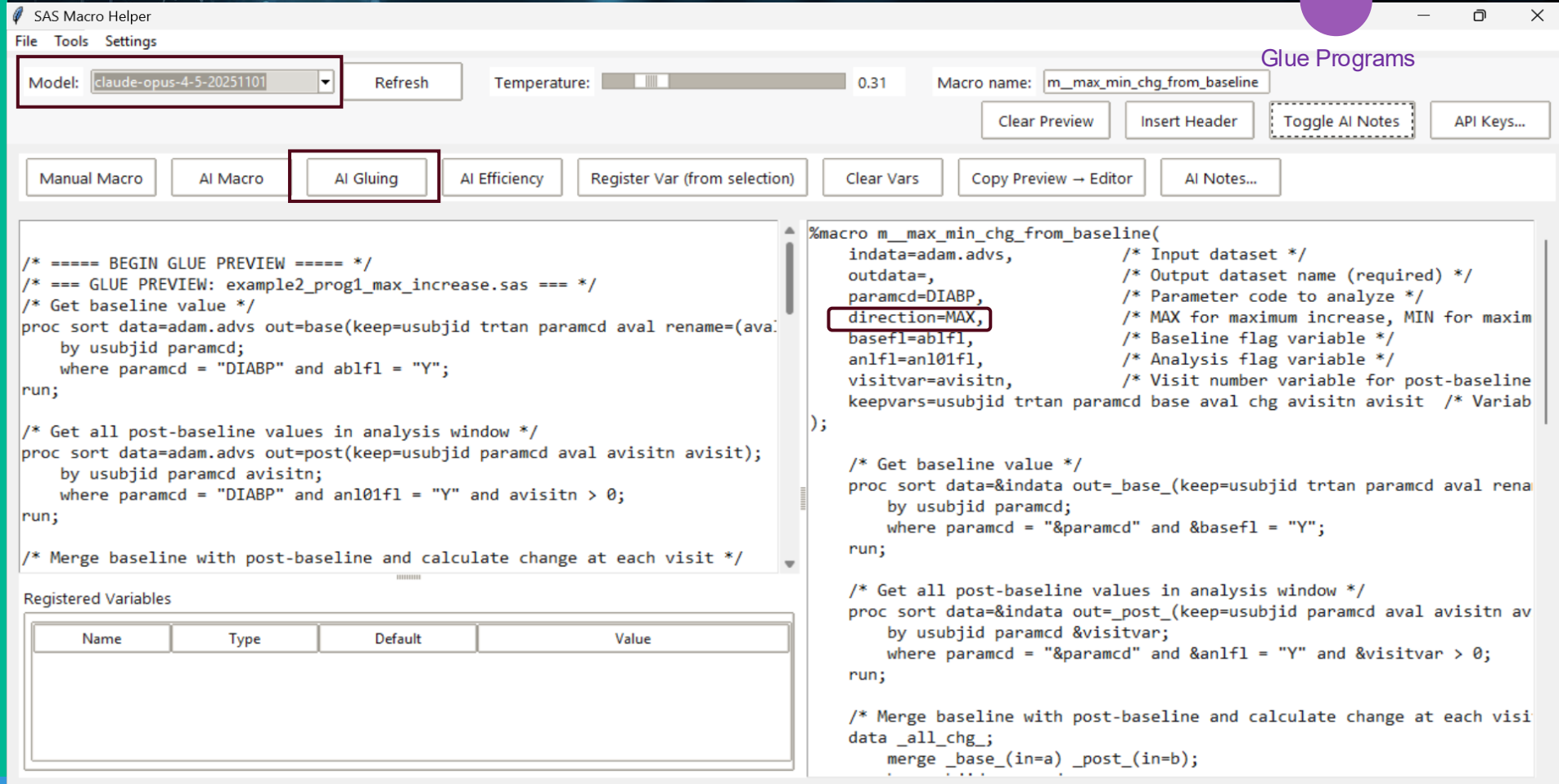
```
verbose code → lean code
```

phastar

SAS Macro Helper

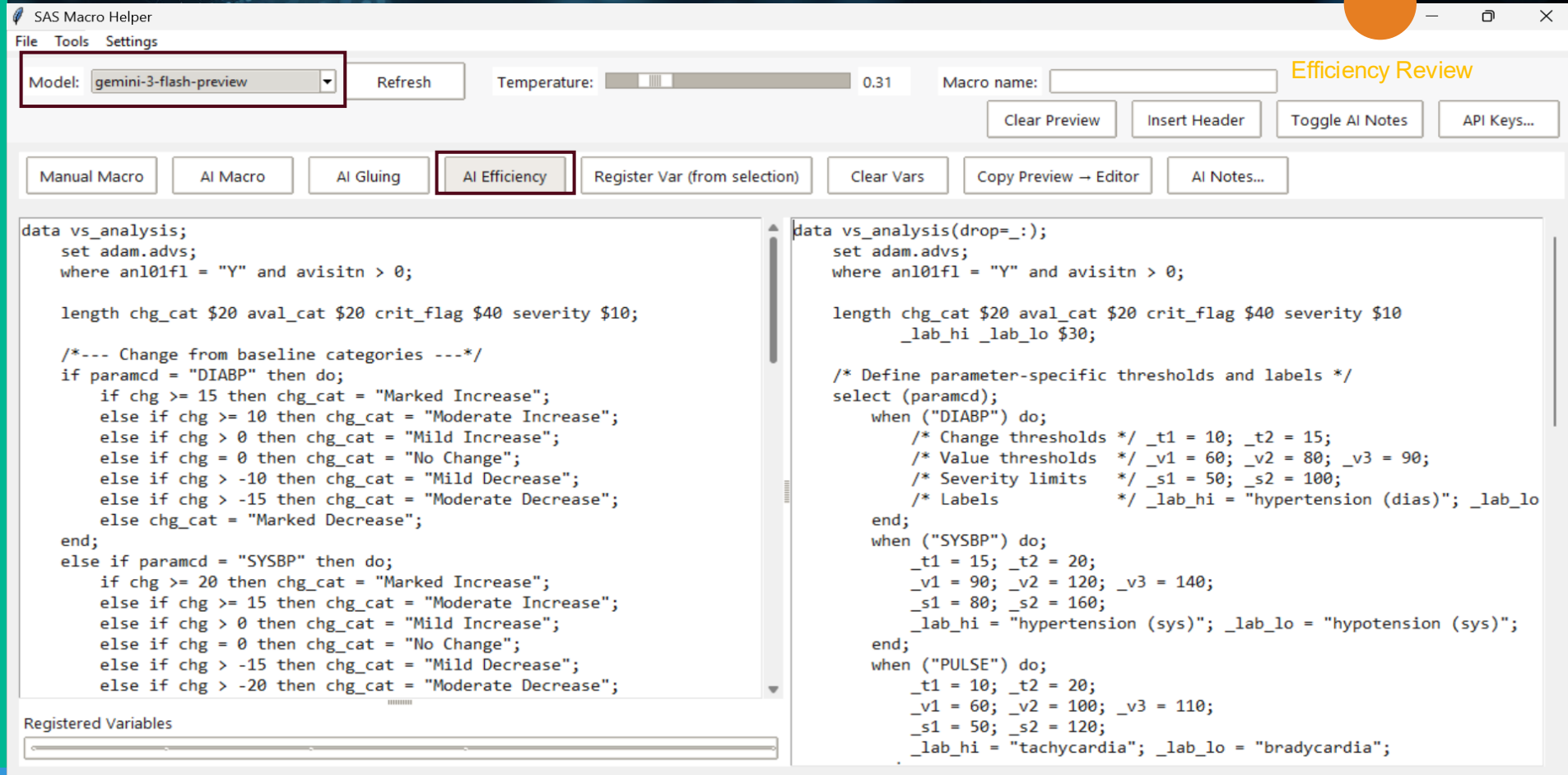


SAS Macro Helper



Glue Programs

SAS Macro Helper



Category 2: Document Organizer

THE PROBLEM

Managing multiple studies means constantly navigating different directory structures to find key documents. Time-consuming and easy to lose track.

THE SOLUTION

A simple tool that creates a centralized folder of shortcuts to all relevant study documents, organized by category.

File & Folder Shortcuts

Auto-Categorization

Batch Operations

Save & Share

Built entirely through iterative prompting - no API integration required.

phastar

The Iterative Journey

From first prompt to polished tool - a real development session that lasted (in total) ~3h

1

Initial Prompt → Working Application

“Help me design a Python application to collect shortcuts...”

Result: ~650 lines of working Python with full GUI

2

Feature Request: Subfolder Toggle

“I’d like the option to include/exclude subfolders in batch scan”

Result: Checkbox added — 2 minutes

3

Real-World Edge Case: Temp Files

“I’m seeing ~$ADAE files that don't exist” (Office lock files)

Result: Filter added — 1 minute

4

Feature Expansion: Custom Categories

“Enable users to add custom categories”

Result: Category management dialog — 5 minutes

phastar

Learning as You Go

After the AI generates code, you can read it, understand patterns, and extend it yourself.

TRADITIONAL

1. Learn syntax
2. Study fundamentals
3. Practice exercises
4. Eventually build useful tools

THIS APPROACH

1. Describe what you need
2. Get working code
3. Read and understand it
4. Modify and extend it

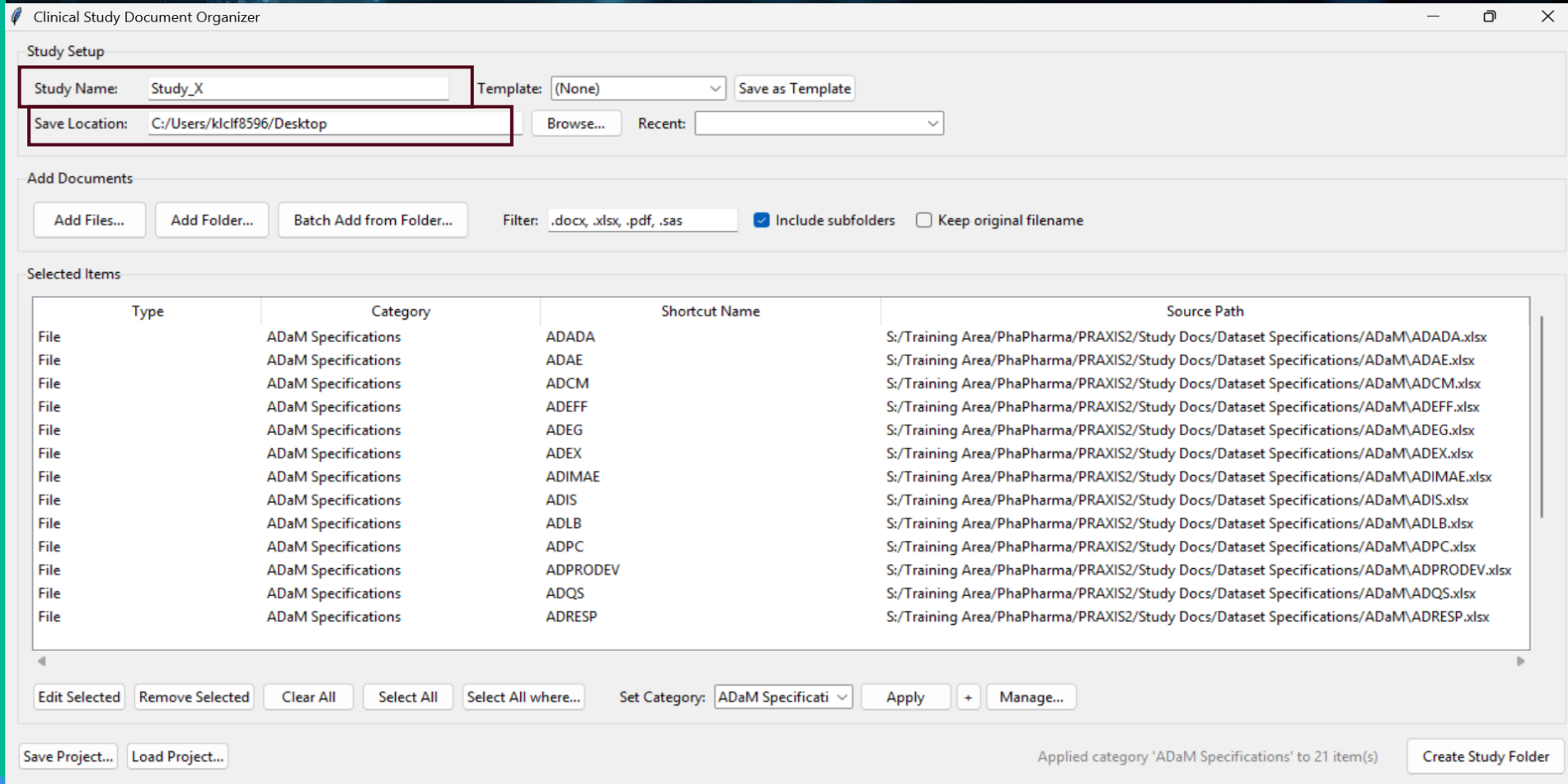
Real Example: Features I Added Myself

"Select All" and "Select All where…" button/functionality — learned from existing patterns

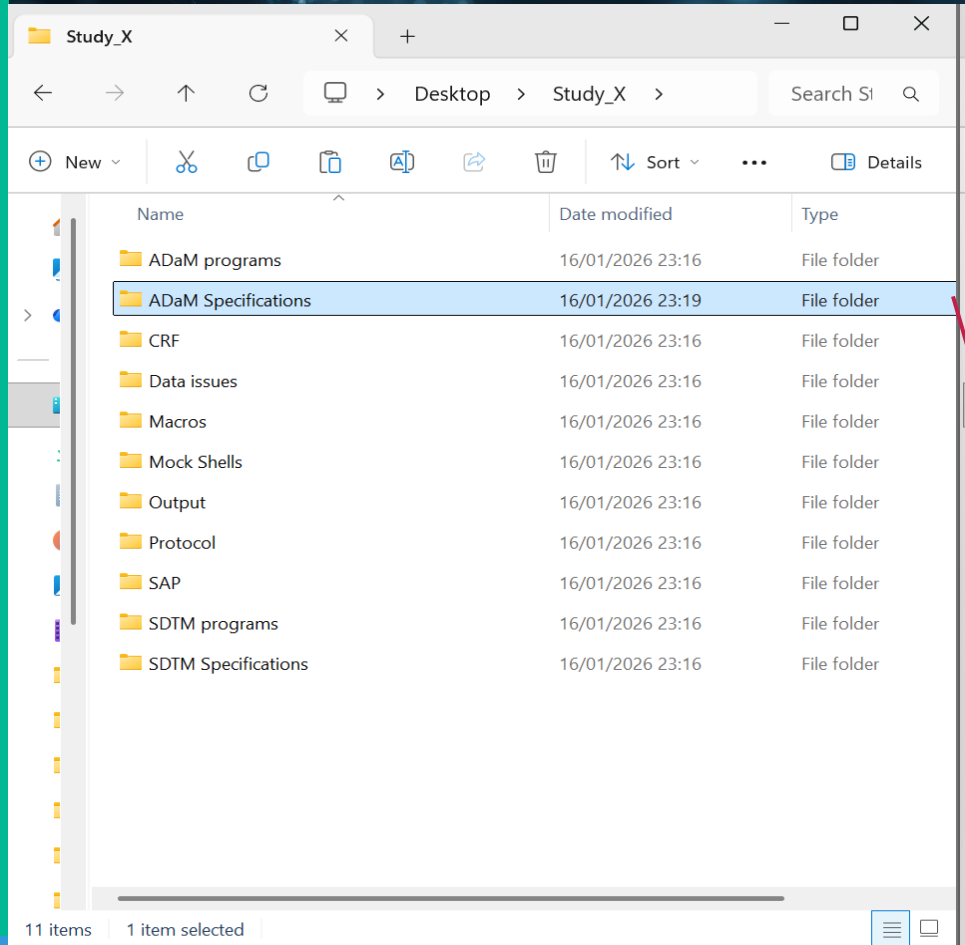
"Keep original filename" — studied checkbox implementation

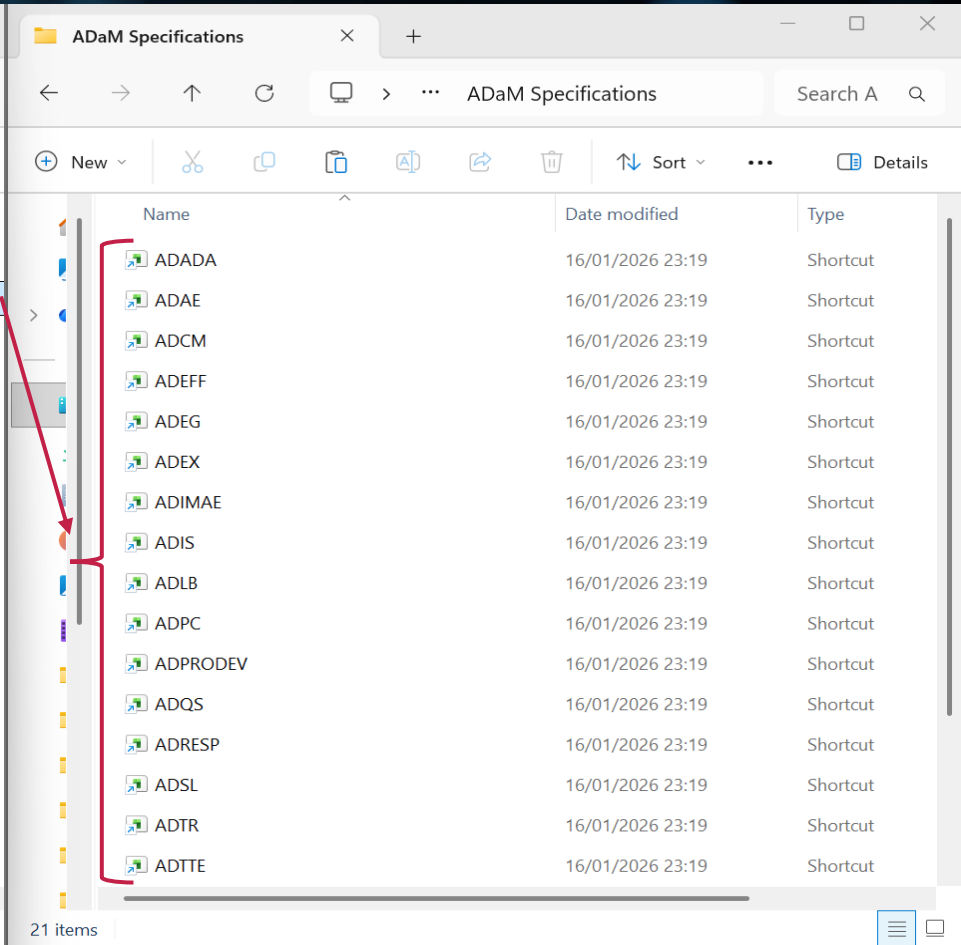
phastar

Document Organizer



Document Organizer





Which Approach When?

API-Based Tools

Choose when you need:

✓ Precise control over AI behavior
✓ Integration with existing systems
✓ Reproducible, consistent outputs
✓ Choice of multiple models
✓ Domain-specific prompting

Requirements:
→ API key(s) and costs
→ More initial setup
→ Some programming knowledge helps

Prompt-Based Tools

Choose when you need:

✓ Quick solution to a specific problem
✓ Standalone utility tool
✓ Iterative refinement through testing
✓ No ongoing API costs
✓ Learning opportunity

Requirements:
→ Access to an LLM (e.g., Claude)
→ Clear problem description
→ Willingness to test and iterate

phastar

Key Takeaways

01

You can build this yourself

No deep programming expertise required. Describe the problem, iterate through testing.

02

LLMs are practical for real work

These tools can solve genuine productivity problems in clinical programming.

03

Choose your approach wisely

API integration for precision; prompt-based for quick, standalone utilities.

04

It’s also a learning pathway

Start with working code, understand it, extend it manually.

phastar

Thank You

Questions?

Dr Damian Wierzbicki
Statistical Programmer, Phastar
damian.wierzbicki@phastar.com