

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



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.

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.

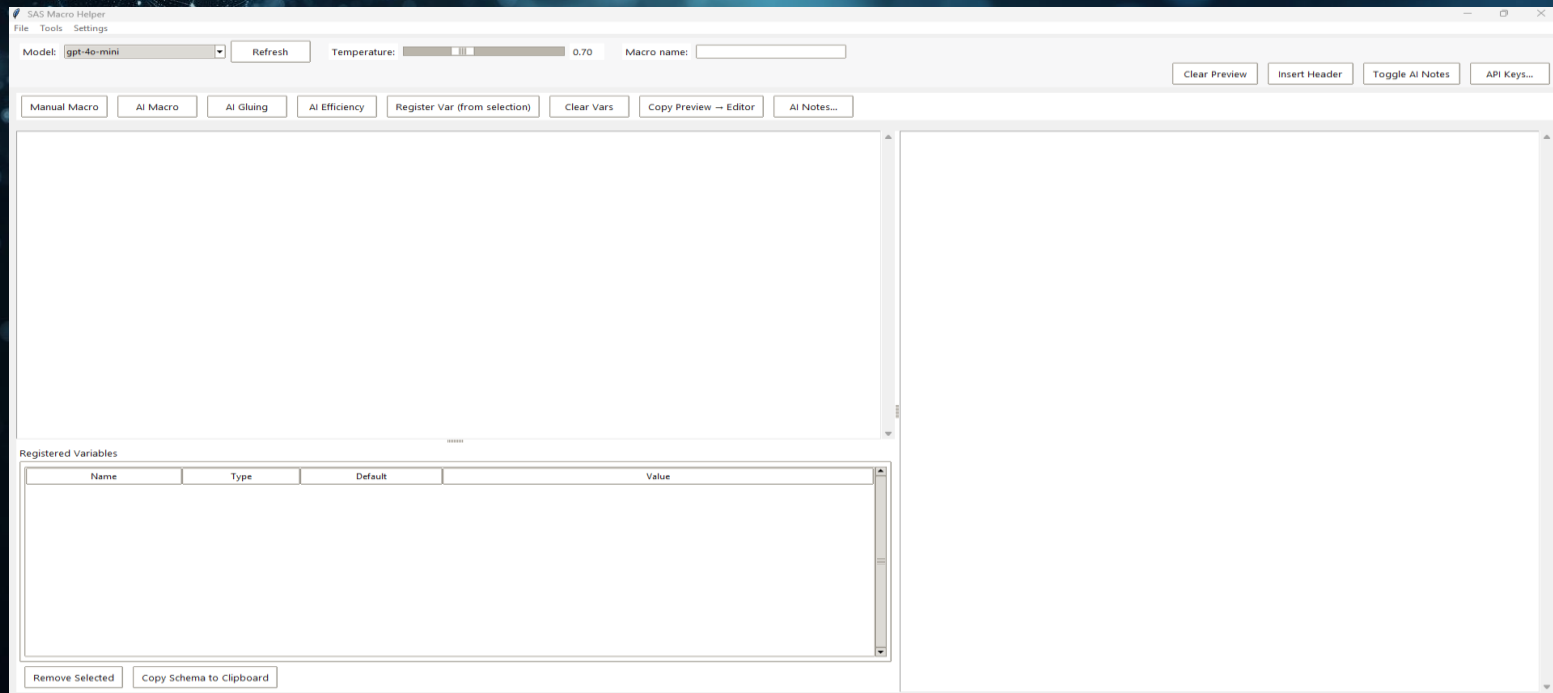
Example

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.



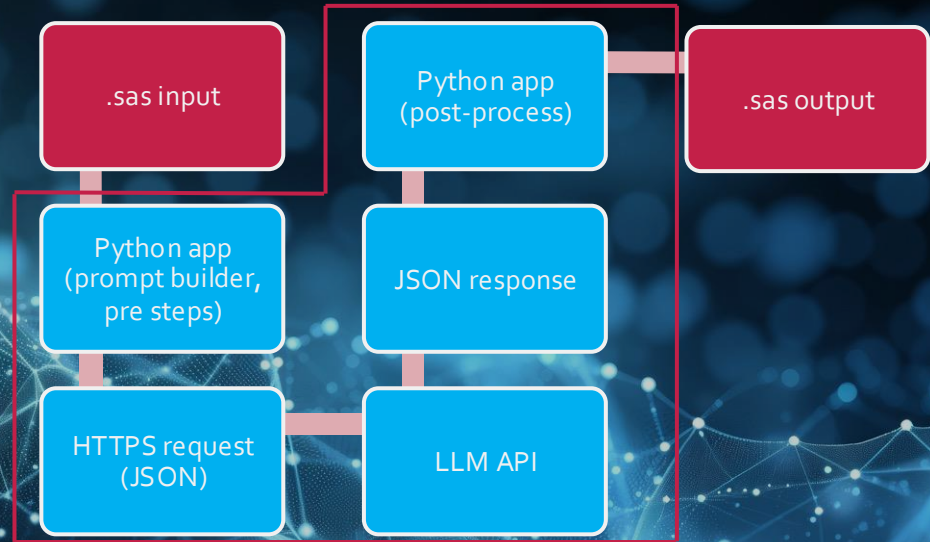
Programmatic, not chat: how the API works

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.



Handled by the tool!

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

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?

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
```


SAS Macro Helper

Convert to Macro

SAS Macro Helper

File Tools Settings

Model: gpt-5.2

Refresh

Temperature:

1.00

Macro name:

m_maxchg_from_baseline

Clear Preview

Insert Header

Toggle AI Notes

API Keys...

Manual Macro

AI Macro

AI Gluing

AI Efficiency

Register Var (from selection)

Clear Vars

Copy Preview → Editor

AI Notes...

```
/* Get baseline value */
proc sort data=adam.advs out=base(keep=usubjid trtan paramcd aval rename=(aval:
  by usubjid paramcd;
  where paramcd = "DIABP" and ablfl = "Y" ;
run;

/* Get all post-baseline values */
proc sort data=adam.advs out=post(keep=usubjid paramcd aval avisitn avisitn);
  by usubjid paramcd avisitn;
  where paramcd = "DIABP" and anl01fl = "Y" and avisitn > 0;
run;

/* Merge and calculate change at each visit */
data all_chg;
  merge base(in=a) post(in=b);
  by usubjid paramcd;
  if a and b;
```

Registered Variables

Name	Type	Default	Value
datain	DATASET		adam.advs
param_name	STRING		DIABP
dataout	DATASET		diabp_maxchg

```
*****
Program Name      : example1_maxchg_from_baseline.sas
Type              : Macro / Program
Description       : Generated by SAS Macro Helper
Author           : klclf8596
Date Created      : 2026-01-16
Files Used        : (none)
Status           : Draft
*****

%macro m_maxchg_from_baseline(datain=, param_name=, dataout=);
%local _param_up;
%let _param_up=%upcase(%superq(param_name));

proc sort data=&datain out=_mmax_base(keep=usubjid trtan paramcd aval rena
  by usubjid paramcd;
  where upcase(paramcd)= "&_param_up" and upcase(ablfl)="Y";
run;

proc sort data=&datain out=_mmax_post(keep=usubjid paramcd aval avisitn av
  by usubjid paramcd avisitn;
  where upcase(paramcd)= "&_param_up" and upcase(anl01fl)="Y" and avisitn>
run;

data _mmax_allchg;
  merge _mmax_base(in=a) _mmax_post(in=b);
  ...
```


File Tools Settings

Refresh

Temperature:

0.31

Macro name: m_max_min_chq_from_baseline

[Clear Preview](#)

Insert Header

Toggle AI Notes

API Keys...

Manual Macro

AI Macro

Al Gluing

AI Efficiency

Register Var (from selection)

Clear Vars

Copy Preview → Editor

AI Notes...

```

/* ===== BEGIN GLUE PREVIEW ===== */
/* === GLUE PREVIEW: example2_prog1_max_increase.sas === */
/* Get baseline value */
proc sort data=adam.advs out=base(keep=usubjid trtn paramcd aval rename=(aval
    by usubjid paramcd;
    where paramcd = "DIABP" and ablfl = "Y";
run;

/* Get all post-baseline values in analysis window */
proc sort data=adam.advs out=post(keep=usubjid paramcd aval avisitn avisit);
    by usubjid paramcd avisitn;
    where paramcd = "DIABP" and anl01fl = "Y" and avisitn > 0;
run;

/* Merge baseline with post-baseline and calculate change at each visit */

```

Registered Variables

Name	Type	Default	Value
------	------	---------	-------

```
%macro m_max_min_chg_from_baseline(
    indata=adam.advs,          /* Input dataset */
    outdata=,                  /* Output dataset name (required) */
    paramcd=DIABP,             /* Parameter code to analyze */
    direction=MAX,             /* MAX for maximum increase, MIN for maxim
    basefl=ablfl,              /* Baseline flag variable */
    anlfl=anl01fl,             /* Analysis flag variable */
    visitvar=avisitn,          /* Visit number variable for post-baseline
    keepvars=usubjid trtan paramcd base aval chg avisitn avisit /* Variab
);

/* Get baseline value */
proc sort data=&indata out=_base_(keep=usubjid trtan paramcd aval rena
    by usubjid paramcd;
    where paramcd = "&paramcd" and &basefl = "Y";
run;

/* Get all post-baseline values in analysis window */
proc sort data=&indata out=_post_(keep=usubjid paramcd aval avisitn av
    by usubjid paramcd &visitvar;
    where paramcd = "&paramcd" and &anlfl = "Y" and &visitvar > 0;
run;

/* Merge baseline with post-baseline and calculate change at each visi
data _all_chg_;
    merge _base_ (in=a) _post_ (in=b);
```

SAS Macro Helper

SAS Macro Helper

File Tools Settings

Model: gemini-3-flash-preview Refresh Temperature: 0.31 Macro name: Efficiency Review

Clear Preview Insert Header Toggle AI Notes API Keys...

Manual Macro AI Macro AI Gluing **AI Efficiency** Register Var (from selection) Clear Vars Copy Preview → Editor AI Notes...

```
data vs_analysis;
  set adam.advs;
  where anl01fl = "Y" and avisitn > 0;

  length chg_cat $20 aval_cat $20 crit_flag $40 severity $10;

  /*--- Change from baseline categories ---*/
  if paramcd = "DIABP" then do;
    if chg >= 15 then chg_cat = "Marked Increase";
    else if chg >= 10 then chg_cat = "Moderate Increase";
    else if chg > 0 then chg_cat = "Mild Increase";
    else if chg = 0 then chg_cat = "No Change";
    else if chg > -10 then chg_cat = "Mild Decrease";
    else if chg > -15 then chg_cat = "Moderate Decrease";
    else chg_cat = "Marked Decrease";
  end;
  else if paramcd = "SYSBP" then do;
    if chg >= 20 then chg_cat = "Marked Increase";
    else if chg >= 15 then chg_cat = "Moderate Increase";
    else if chg > 0 then chg_cat = "Mild Increase";
    else if chg = 0 then chg_cat = "No Change";
    else if chg > -15 then chg_cat = "Mild Decrease";
    else if chg > -20 then chg_cat = "Moderate Decrease";
  end;
```

Registered Variables

```
data vs_analysis(drop=_:);
  set adam.advs;
  where anl01fl = "Y" and avisitn > 0;

  length chg_cat $20 aval_cat $20 crit_flag $40 severity $10
        _lab_hi _lab_lo $30;

  /* Define parameter-specific thresholds and labels */
  select (paramcd);
    when ("DIABP") do;
      /* Change thresholds */ _t1 = 10; _t2 = 15;
      /* Value thresholds */ _v1 = 60; _v2 = 80; _v3 = 90;
      /* Severity limits */ _s1 = 50; _s2 = 100;
      /* Labels */ _lab_hi = "hypertension (dias)"; _lab_lo
    end;
    when ("SYSBP") do;
      _t1 = 15; _t2 = 20;
      _v1 = 90; _v2 = 120; _v3 = 140;
      _s1 = 80; _s2 = 160;
      _lab_hi = "hypertension (sys)"; _lab_lo = "hypotension (sys)";
    end;
    when ("PULSE") do;
      _t1 = 10; _t2 = 20;
      _v1 = 60; _v2 = 100; _v3 = 110;
      _s1 = 50; _s2 = 120;
      _lab_hi = "tachycardia"; _lab_lo = "bradycardia";
```

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.

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

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

Document Organizer

Clinical Study Document Organizer

Study Setup

Study Name: Study_X

Template: (None)

Save as Template

Save Location: C:/Users/klclf8596/Desktop

Browse...

Recent:

Add Documents

Add Files...

Add Folder...

Batch Add from Folder...

Filter: .docx, .xlsx, .pdf, .sas

☒ Include subfolders

☐ Keep original filename

Selected Items

Type	Category	Shortcut Name	Source Path
File	ADaM Specifications	ADADA	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADADA.xlsx
File	ADaM Specifications	ADAE	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADAE.xlsx
File	ADaM Specifications	ADCM	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADCM.xlsx
File	ADaM Specifications	ADEFF	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADEFF.xlsx
File	ADaM Specifications	ADEG	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADEG.xlsx
File	ADaM Specifications	ADEX	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADEX.xlsx
File	ADaM Specifications	ADIMAE	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADIMAE.xlsx
File	ADaM Specifications	ADIS	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADIS.xlsx
File	ADaM Specifications	ADLB	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADLB.xlsx
File	ADaM Specifications	ADPC	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADPC.xlsx
File	ADaM Specifications	ADPRODEV	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADPRODEV.xlsx
File	ADaM Specifications	ADQS	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADQS.xlsx
File	ADaM Specifications	ADRESP	S:/Training Area/PhaPharma/PRAXIS2/Study Docs/Dataset Specifications/ADaM\ADRESP.xlsx

Edit Selected

Remove Selected

Clear All

Select All

Select All where...

Set Category:

ADaM Specificati

Apply

+

Manage...

Save Project...

Load Project...

Applied category 'ADaM Specifications' to 21 item(s)

Create Study Folder

Document Organizer

Study_X

Desktop > Study_X

Search Study_X

New, Cut, Copy, Paste, Share, Delete, Sort, Details

Name	Date modified	Type
ADaM programs	16/01/2026 23:16	File folder
ADaM Specifications	16/01/2026 23:19	File folder
CRF	16/01/2026 23:16	File folder
Data issues	16/01/2026 23:16	File folder
Macros	16/01/2026 23:16	File folder
Mock Shells	16/01/2026 23:16	File folder
Output	16/01/2026 23:16	File folder
Protocol	16/01/2026 23:16	File folder
SAP	16/01/2026 23:16	File folder
SDTM programs	16/01/2026 23:16	File folder
SDTM Specifications	16/01/2026 23:16	File folder

11 items | 1 item selected

ADaM Specifications

ADaM Specifications

Search ADaM Specifications

New, Cut, Copy, Paste, Share, Delete, Sort, Details

Name	Date modified	Type
ADADA	16/01/2026 23:19	Shortcut
ADAE	16/01/2026 23:19	Shortcut
ADCM	16/01/2026 23:19	Shortcut
ADEFF	16/01/2026 23:19	Shortcut
ADEG	16/01/2026 23:19	Shortcut
ADEX	16/01/2026 23:19	Shortcut
ADIMAE	16/01/2026 23:19	Shortcut
ADIS	16/01/2026 23:19	Shortcut
ADLB	16/01/2026 23:19	Shortcut
ADPC	16/01/2026 23:19	Shortcut
ADPRODEV	16/01/2026 23:19	Shortcut
ADQS	16/01/2026 23:19	Shortcut
ADRESP	16/01/2026 23:19	Shortcut
ADSL	16/01/2026 23:19	Shortcut
ADTR	16/01/2026 23:19	Shortcut
ADTTE	16/01/2026 23:19	Shortcut

21 items

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

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.

Thank You

Questions?

Dr Damian Wierzbicki

Statistical Programmer, Phastar
damian.wierzbicki@phastar.com

