



Real-Time AI-Driven TLF Generation with GAutoTLF

Hao Xu

Tyler Rowsell

AstraZeneca Oncology Biometrics

PHUSE US Connect 2026

GAutoTLF

BETA

Guided Automation of Tables, Listings, and Figures

The Challenge – Traditional TLF generation



Highly manual
workflow



Constant
rework for
changing scope



Study-specific
knowledge
requirements

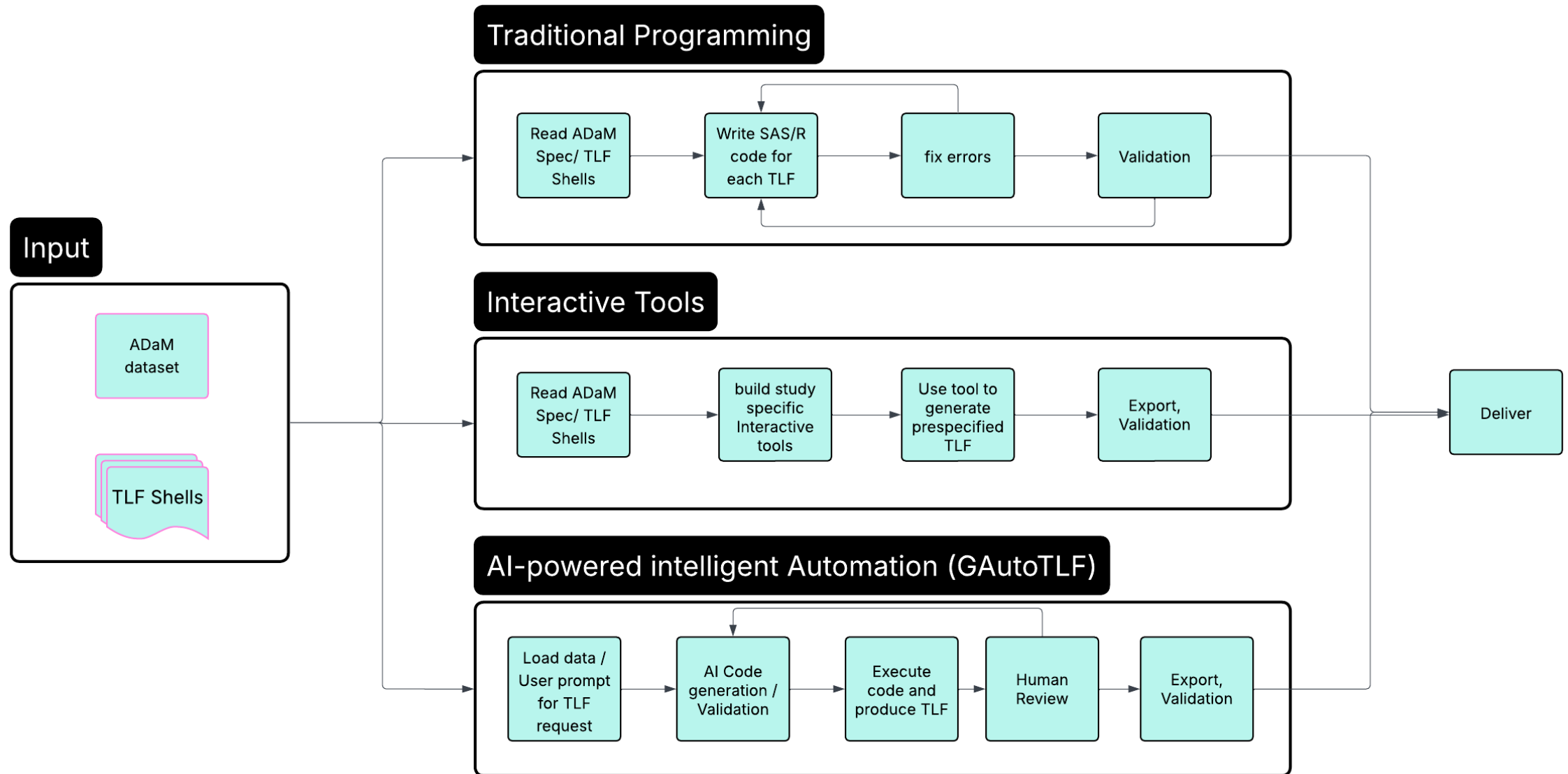


Intensive
double-
programming
validation



Slow
turnaround

Comparison of TLF Generation Approaches



What is GAutoTLF

Guided Automation of Tables, Listings & Figures

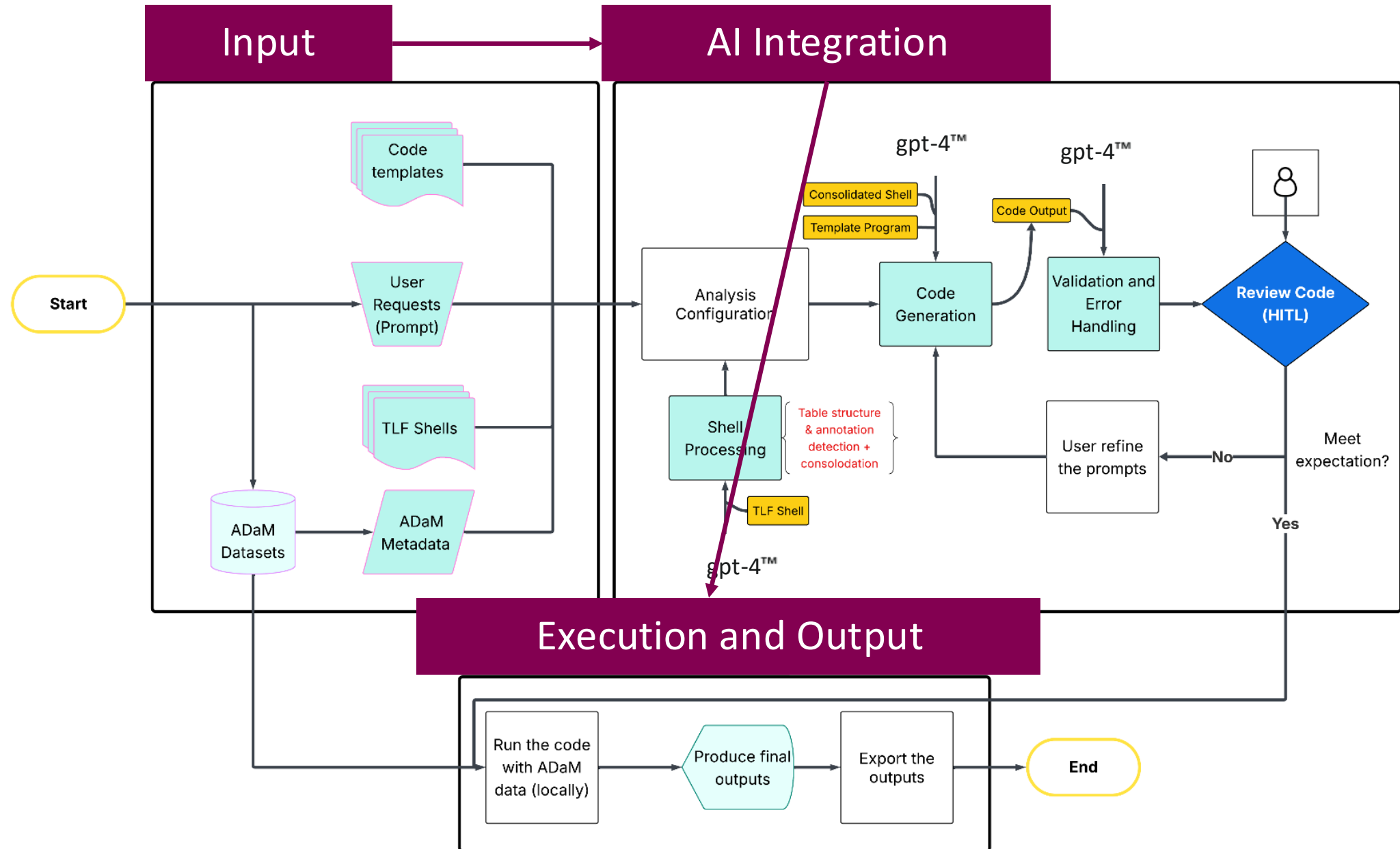
- An AI-Powered solution to automate clinical TLF generation

by combining:

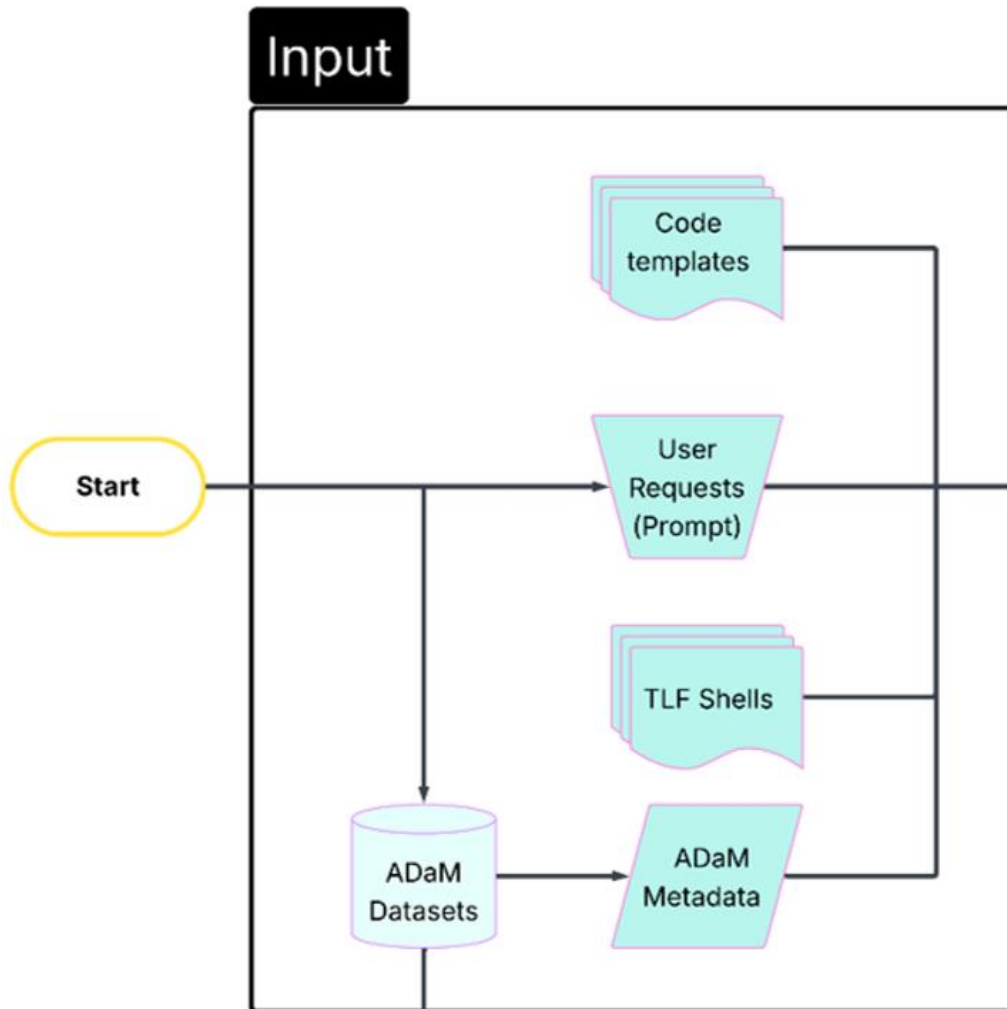
-  Standard TLF shells
-  Template code library
-  Study ADaM metadata
-  Natural language prompts



GAutoTLF workflows



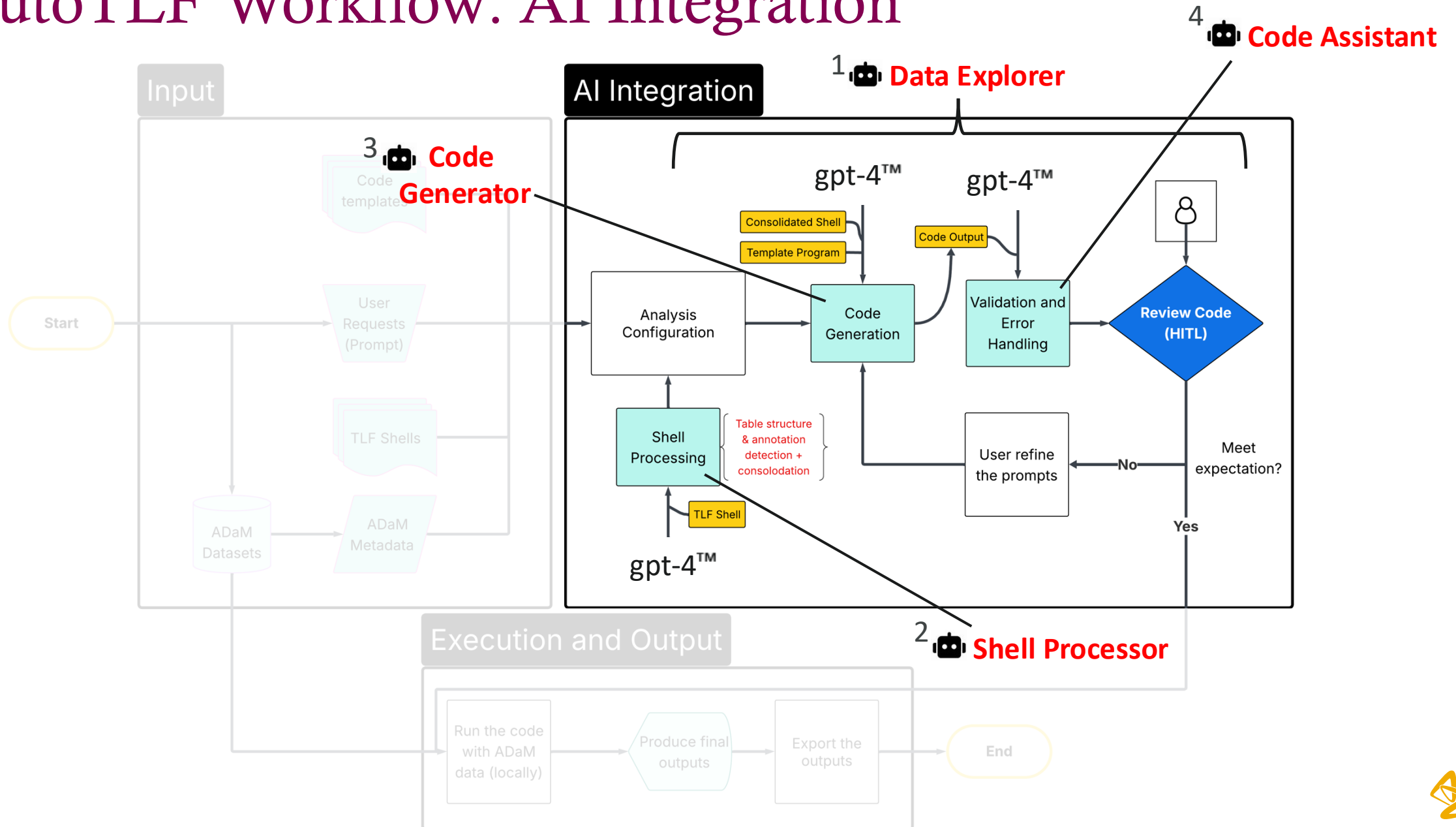
GAutoTLF Workflow: Input



- **Consolidate inputs:** ADaM metadata (variable names, domains, populations) + user requests
- **Align to standards:**
 - map those inputs to TLF shell templates.
 - Confirm which datasets, variables, and values are required.
 - only metadata is shared with AI.
- **Configure for AI:** create a clear, compliant configuration

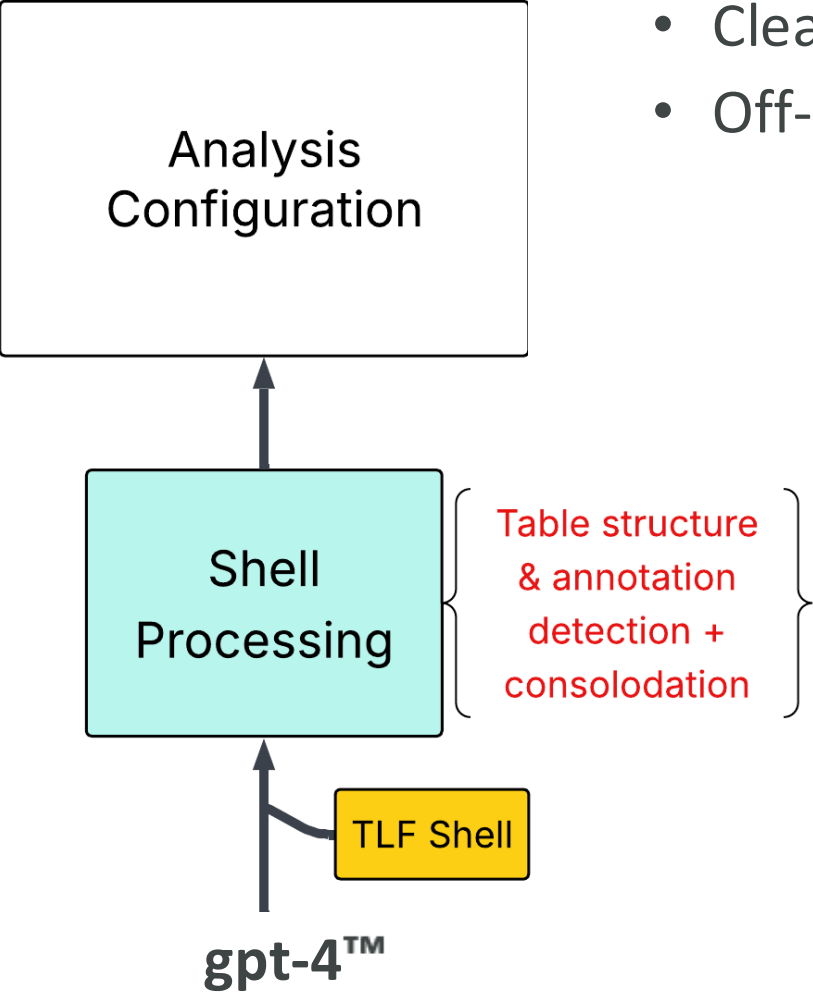


GAutoTLF Workflow: AI Integration



Shell Processor

- Clear, unambiguous communication of analysis requirements
- Off-loading interpretation workload from code generation step



Input

 **TM**
TLF Shell

ADSL (FASFL = Y)		ARM			
Demographics		Arm A N = xxx	Arm B N = xxx	Total N = xxx	
	Statistic				
Age (Years)	n	xxx	xxx	xxx	AGE
	Mean	xx.x	xx.x	xx.x	
	SD	x.x	x.x	x.x	
	Median	xx.x	xx.x	xx.x	
	Min	xx	xx	xx	
Max	xxx	xxx	xxx		
Age group (Years)					AGEGR
<50	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
>= 50 - < 65	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
>= 65 - < 75	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
>= 75	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
Missing	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
Sex					SEX
Female	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	
Male	n (%)	xx (xx.x)	xx (xx.x)	xx (xx.x)	

*annotated

Output

 **Array** ➡  **Dataframe**

row	col	value	type
1	1	ADSL	DATASET
1	1	FASFL = Y	FILTER
...	...	...	...

+ formatted
markdown shell 



Code Generator

- Context-aware guided code generation for selected TLF
- Current scope: core safety outputs (40)

Input



Queried dataframe text

+



Markdown shell

AZSOL Template Code*

```
vars <- c("[analysis var 1]", "[analysis var 2]", "[analysis var 3]",
"[analysis var 4]", "[analysis var etc.]")

var_labels <- c(
  "[analysis label 1]",
  "[analysis label 2]",
  "[analysis label 3]",
  "[analysis label 4]",
  "[analysis label etc.]"
)

result <- basic_table(show_colcounts = TRUE) %>%
  split_cols_by(var = "[treatment var]") %>%
  add_overall_col("Total") %>%
  analyze_vars(
    vars = vars,
    var_labels = var_labels
  ) %>%
  build_table(adsl)

result
```

User free-text prompt

Output



Executable **R code** for
specified analysis
configuration



Table Output:

Table 14.1.x
Demographics (full analysis set)

	Placebo (N=86)	Xanomeline High Dose (N=84)	Xanomeline Low Dose (N=84)	Total (N=254)
Age				
n	86	84	84	254
Mean (SD)	73.2 (8.4)	74.4 (7.9)	73.7 (8.3)	73.1 (8.2)
Median	76.0	76.0	77.5	77.0
Min-Max	52.0-89.0	56.0-88.0	51.0-88.0	51.0-89.0
Pooled Age Group 1				
n	86	84	84	254
<65	14 (16.3%)	11 (13.1%)	8 (9.5%)	33 (12.9%)
>65	30 (34.9%)	18 (21.4%)	29 (34.5%)	77 (30.3%)
65-80	42 (48.8%)	55 (65.5%)	47 (56%)	144 (56.7%)
Sex				
n	86	84	84	254
F	53 (61.6%)	40 (47.6%)	50 (59.5%)	143 (56.3%)

[Download Data \(xlsx, csv, rgt\)](#) [Download as HTML](#)

gpt-4™

Consolidated Shell

Template Program

Code
Generation



Template Code Example

```
vars <- c("[analysis var 1]", "[analysis var 2]", "[analysis var 3]", "[analysis var 4]", "[analysis var etc.]")
var_labels <- c(
  "[analysis label 1]",
  "[analysis label 2]",
  "[analysis label 3]",
  "[analysis label 4]",
  "[analysis label etc.]"
)
result <- basic_table(show_colcounts = TRUE) %>%
  split_cols_by(var = "[treatment var]") %>%
  add_overall_col("Total") %>%
  analyze_vars(
    vars = vars,
    var_labels = var_labels
  ) %>%
  build_table(ads1)
result
```

- Placeholders,
 - dynamically populated by LLM
 - Static foundational/structural code



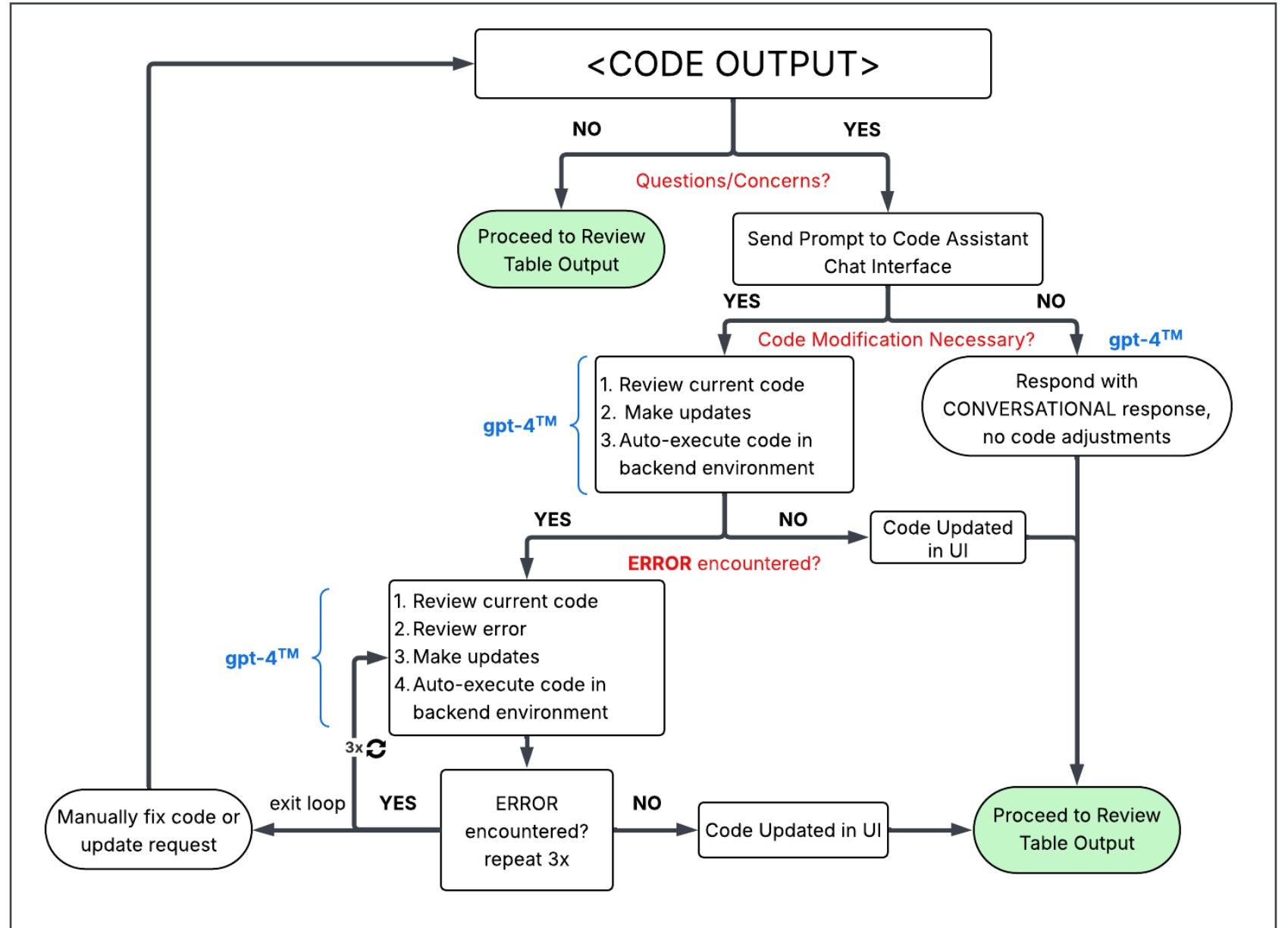
Code Assistant

– Validation and Error Handling

gpt-4™

Code Output

Validation and
Error
Handling



Data Explorer

Purpose: Rapid **ADaM** data **investigation** and visualization using **natural language**

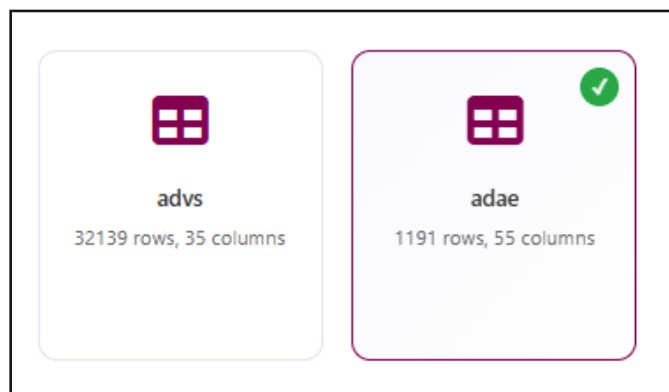
1. Inform analysis configuration of standard TLF output.

e.g., “show me all the unique values of AVAL when PARAMCD = ‘xx’”

2. Generate rapid biometrics-related insights regarding ADaM data to inform study team discussion or programming-specific decisions.

e.g., “are there any subjects that meet [complex criteria]”

Select Data

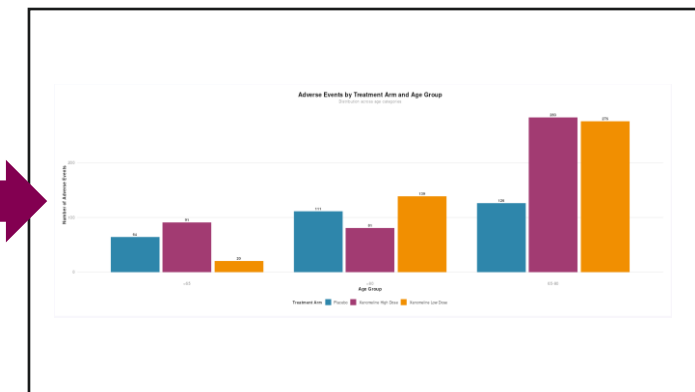


Inquiry

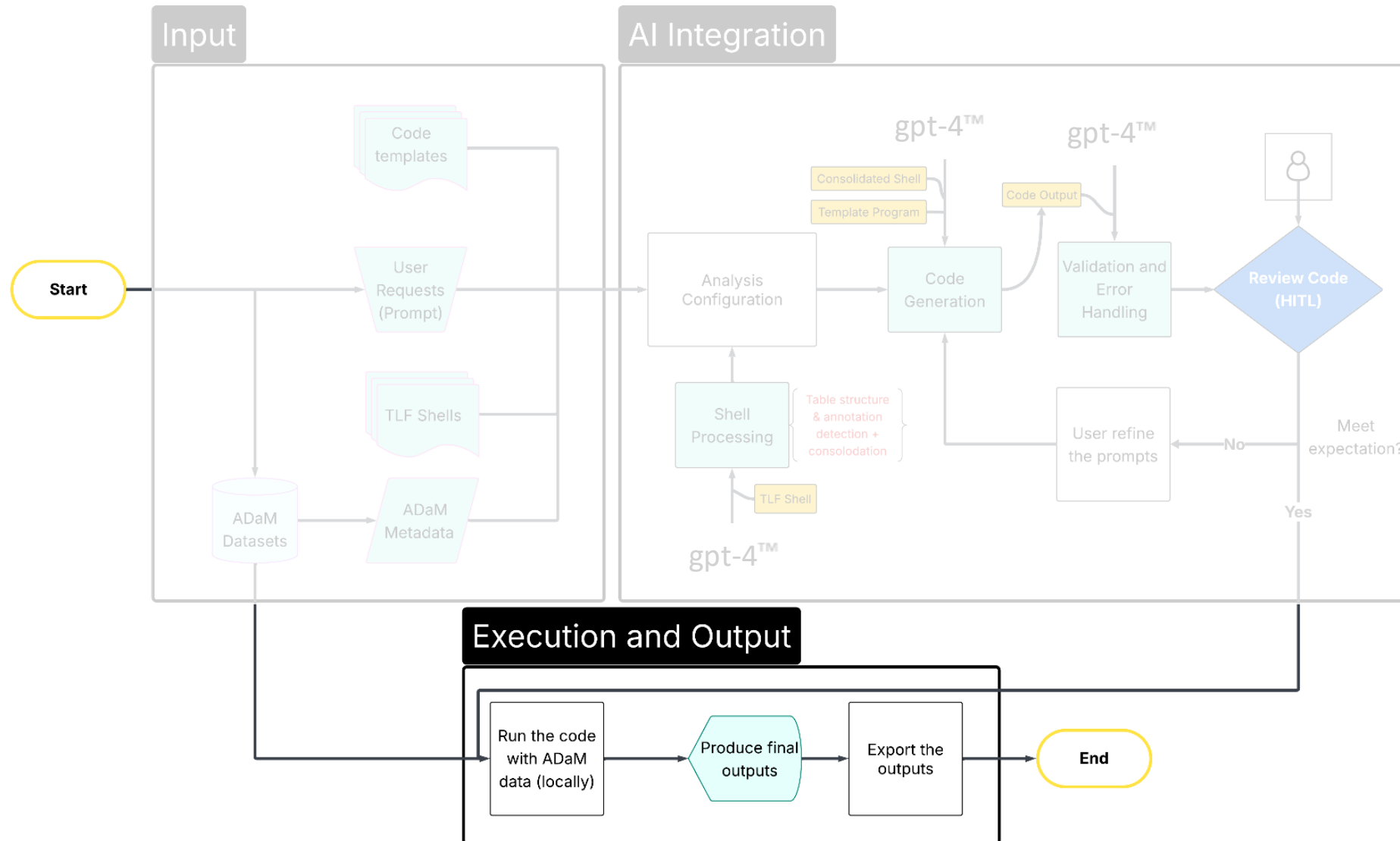
How many AEs in each treatment arm by age group?

Generate R Code

Run Code & View Results



GAutoTLF Workflow: Execution and Output



Execution and Output

Input



Dataset Metadata



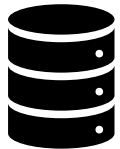
adae

1191 rows, 55 columns



adsl

254 rows, 49 columns



Subject Data

Code Generation

API

gpt-4™

```
1 vars <- c("AGE", "AGEGR1", "SEX", "RACE", "ETHNIC")
2 var_labels <- c(
3   "Age",
4   "Pooled Age Group 1",
5   "Sex",
6   "Race",
7   "Ethnicity"
8 )
9
10 vars_continuous <- c("AGE")
11 vars_categorical <- c("AGEGR1", "SEX", "RACE", "ETHNIC")
12
13 adsl <- adsl %>%
14   rename_with(toupper) %>%
15   mutate(across(
16     .cols = all_of(vars),
17     .fns = ~ ifelse(. == "" | is.na(.), NA, .))
18   ) %>%
19   mutate(across(where(is.character), as.factor)) %>%
20   filter(toupper(FASFL) == "Y") %>%
21   mutate(TRT01P = factor(TRT01P, exclude = ""))
22
23 adsl <- adsl %>%
24   rename_with(toupper) %>%
25   mutate(across(
26     .cols = all_of(vars_categorical),
27     .fns = ~ fct_explicit_na(as.factor(.x), na_level = "MISSING")
28   ))
29
30 result <- basic_table(show_colcounts = TRUE,
31   title = "Table 14.1.x",
32   subtitle = "Demographics (Full analysis set)",
33   main_footer = c("n Number of subjects in analysis for a continuous variable and
34     ..."))
```

Run Code

Code Execution & Output

Shiny App

Analysis Output

Table Output:

Table 14.1.x
Demographics (Full analysis set)

	Placebo (N=86)	Xanomeline High Dose (N=84)
Age		
n	86	84
Mean (SD)	75.2 (8.6)	74.4 (7.9)
Median	76.0	76.0
Min - Max	52.0 - 89.0	56.0 - 88.0
Pooled Age Group 1		
n	86	84
<65	14 (16.3%)	11 (13.1%)
>80	30 (34.9%)	18 (21.4%)
65-80	42 (48.8%)	55 (65.5%)
Sex		
n	86	84
F	53 (61.6%)	40 (47.6%)

Download Data (xlsx, csv, xpt)

Download as HTML

Download as: XLSX, CSV, XPT, HTML



Demo

Speaker: Tyler Rowsell



 **GAutoTLF** BETA [What is GAutoTLF?](#) [Code Generation](#)

 **Data Upload Center**
Upload your ADaM data and configure your analysis


Drag and drop your files here or click to browse

Browse... No file selected

Excel CSV XPT SAS RData ZIP

Please upload data files to begin analysis.

 **Analysis Configuration**
Configure your analysis parameters

Model to use:
gpt-4-32k

☒ Use available AZ templates
☐ Exploratory analysis

Select AZSOL template:
Select an option

☐ Add shell details

Describe the request: ⓘ

e.g., toxicity grade >= 3, use PARAMCD = 'DUREXP', use X variable instead of Y, etc.



Current GAutoTLF Coverage

Scope & Maturity

- 40 standard TLF templates aligned with AZ Standard Output Library (AZSOL)

Covered Domains

- Study Population & Demographics
- Exposure
- Adverse Events
- Laboratory Results & Vital Signs
- Listings



Pilot Results

Data Source	Applicable Outputs	Success Rate	Average Attempts	Average Time per output (mins)
CDISC Dummy data	24	96% (23)	1.2	3.3
Study 1	37	95% (35)	1.5	2.2
Study 2	31	94% (29)	1.5	4
Pooled Safety data	25	92% (23)	1.2	1.5
Overall		94%	1	3



Benefits

Faster Turnaround

- Real-time TLF generation
- Supports rapid responses to emerging clinical questions

Improved Consistency & Quality

- Driven by standardized TLF shells and template library
- Built-in validation + iterative AI refinement process

Free Programmers to focus on strategic work

- Automate routine, repetitive TLF task

Scalable Across Studies & Therapeutic Areas

- Eliminates need for customized tools per study



Future Directions

Broaden Output Coverage

- Expand beyond safety to efficacy, add Figures & additional domains



Next-Gen LLM Enhancements

- Faster, more reliable code generation using upgraded models



Integrate with Internal Platforms

- Seamless links to data storage, analytics, standards, and shell systems



Plugin & Extension Ecosystem

- Custom modules for domain-specific logic



Parameterized Architecture

- More flexible, reusable function-based generation



Move Toward GxP Compliance

- Audit trails, validation framework, traceability



Acknowledgement

Avinash Reddy Pati

for strategic insights and guidance

Project Development Team

Nandini Thampi • Dishank Jani • Jason Yang • Piotr Bugajski

for technical expertise and collaboration

Oncology Biometrics Leadership, especially Mark Reynolds

for sponsorship and support





Q&A

Contact

Hao Xu – hao.xu5@astrazeneca.com

Tyler Rowsell – tyler.rowsell@astrazeneca.com