**Paper ET03**

# Automated Code Generation: Evaluating Deterministic and Probabilistic Approaches

Malan Bosman, Clymb Clinical, Bloemfontein, South Africa

## ABSTRACT

In the last few years, the clinical development industry has experienced rapid changes with new CDISC standards (including the Analysis Results Standard), open-source packages, technology solutions, and Generative AI. As professionals in this industry, we need to make sense of the possibilities and risks, and forge new ways of doing things that are faster, cheaper, and of high quality.

This paper unpacks different approaches for the automated generation of R or SAS programs producing Tables, Figures and Listings (TFLs). Two broad categories of approaches emerge: Deterministic and Probabilistic. The Deterministic approach refers to a system of logic and algorithms that produce consistent results for a given input, while the Probabilistic approach refers to a Generative AI model generating code with variability in the result.

The pros and cons of both approaches are discussed based on demonstrations of each approach. Factors such as implementation requirements, workability, reliability, and success factors are considered. Using the CDISC Analysis Results Standard (ARS) model and associated software tools, solutions are demonstrated for automatically producing R code for TFL generation based on input metadata.

## INTRODUCTION

The clinical development industry is experiencing rapid change, which presents opportunities for dynamic and positive transformation of our procedures and processes. Several key developments have converged to create new possibilities for automation in TFL generation:

The launch of ChatGPT 3.5 in November 2022 brought Generative AI into mainstream awareness and usage. The release of CDISC's Analysis Results Standard (ARS) in April 2024 provided a standardized framework for describing analysis results and associated metadata. Combined with the continued development of open-source packages in R and growing adoption of modern technology solutions, these changes have created an unprecedented opportunity to rethink how we approach TFL automation.

Historically, the clinical development workflow has progressed from Raw data through SDTM (June 2004) to ADaM (December 2009), with TFLs as the final output. With the introduction of ARS in April 2024, we now have a standard that specifically addresses the TFL generation step, enabling new approaches to automation that were previously not possible.

This paper explores two fundamentally different approaches to automated code generation for TFLs: Deterministic and Probabilistic. The Deterministic approach uses explicit, rule-based logic where the same input consistently produces the same output. The Probabilistic approach leverages Generative AI models that may introduce variability in the generated code. Both approaches are evaluated on their implementation requirements, workability, and key success factors.

## BACKGROUND: ANALYSIS RESULTS STANDARD

The Analysis Results Standard (ARS) was created by CDISC with the aim to facilitate automation, reproducibility, reusability, and traceability of analysis results data. This addresses the inefficiency of having no standard way of describing and organizing clinical trial analysis results found in tables and figures. The ARS provides a Logical Data Model that describes analysis results and associated metadata.

The ARS Logical Data Model represents analysis results through a schema showing all components with accompanying metadata linked to a single reporting event. This metadata is divided into model

components, which together describe the transformations that need to be applied to ADaM dataset(s) to produce the results for the reporting event. These results are combined in a flat table with context for each result, known as an Analysis Results Dataset (ARD).

### KEY ARS COMPONENTS

To create results and ultimately an ARD, various model components need to be considered in combination. Four key components are particularly important:

Analysis Set: The subject population to be analyzed.

Data Subset: Conditions describing the selection of data to be included in the analysis.

Analysis Grouping: Characteristics used to cluster the subject populations.

Analysis Method: The statistical operation applied to the data, which consists of one or more Operations.

The metadata for a reporting event is organized into classes with fields that house the metadata elements. Each output contains many analyses, and for each analysis, the automation engine loops through its Analysis Set, Data Subset, Analysis Grouping, and Analysis Method. By processing these components systematically, results are obtained for each analysis, all of which are combined to form the ARD.

## SCOPE OF AUTOMATED CODE GENERATION

The focus of this paper is on the automated generation of code that transforms ADaM datasets into TFLs. More specifically, the scope covers the transformation from ADaM datasets to Analysis Results Datasets (ARDs), which serve as an intermediate step before final TFL rendering. This represents a critical component in the overall TFL automation workflow.

In a complete automation pipeline, ARS metadata defines what analyses need to be performed, automated code generation produces the programs to execute those analyses, ADaM datasets provide the input data, ARDs contain the calculated results, and Analysis Display Metadata (ADM) describes how to format and render the final TFLs. This paper concentrates on the code generation step that bridges ARS metadata to ARD production, using R as the programming language.

## DETERMINISTIC APPROACH

The Deterministic approach to automated code generation relies on explicit, rule-based logic and algorithms that produce consistent results for a given input. This approach is characterized by having no randomness and high reproducibility. Think of it as using functions, macros, and logical operations where the same input always yields the same output.

### IMPLEMENTATION WITH SIERA

The siera package exemplifies the Deterministic approach. siera is an open-source R package that ingests ARS metadata for a reporting event and converts the metadata to produce executable R scripts. These R scripts can be run as-is (combined with the relevant ADaM dataset) to produce each output's ARD.

The workflow operates as follows: ARS metadata defines what needs to be computed, siera generates ARD-ready R scripts based on this metadata, ADaM datasets are referenced by the scripts, and running the scripts produces the Analysis Results Data. This approach avoids the "black box" problem by making the intermediate R code visible and editable.

### INTEGRATION WITH OTHER SOFTWARE

Naturally, software exists in an ecosystem with other tools, each fulfilling specific purposes. For siera, two direct links are required for a complete process:

First, the ARS metadata needs to be populated for a specific reporting event. To populate this metadata, information from the Statistical Analysis Plan, Protocol, TFL Shells, and other sources is required and needs to be combined in a structured way complying with the ARS Logical Data Model. Open-source software fulfilling this purpose includes TFL Designer Community Version, where shells can be designed with appropriate metadata and exported as ARS metadata in JSON or Excel format.

Second, once the ARDs have been created by siera, the final step typically includes producing displays. To produce displays, Analysis Display Metadata (ADM) is needed, which describes the formatting of the final outputs. TFL Designer Enterprise exports ADM, which can be ingested programmatically to combine the ARD with ADM, producing the final TFLs.

#### GETTING STARTED WITH SIERA

To use siera, a fully completed ARS metadata file for a reporting event is used as input. This can be either JSON (the official ARS format) or adapted ARS metadata in Excel format. The package can be installed from CRAN and has been approved by COSA (Coalition for Open-Source in Biostats).

A basic implementation involves the following R code:

library(siera) json_path <- ARS_example("ARS_V1_Common_Safety_Displays.json") output_folder <- tempdir() ADaM_folder <- dirname(normalizePath(ARS_example("ADSL.csv"))) readARS(json_path, output_folder, ADaM_folder)

The result of this call is multiple R programs, one for each output to be generated. Each R script contains references to the ADaM dataset and can be run to produce the ARD. The produced R script is populated with metadata read from the ARS JSON file, including the Analysis Set being applied (e.g., SAFFL = "Y") and the grouping of datasets by treatment variables. By translating these metadata elements into workable R code, the script generates all results for an output and combines them into the ARD.

## PROBABILISTIC APPROACH

The Probabilistic approach to automated code generation leverages Generative AI models to produce code. This approach is characterized by statistical inference and prediction-based generation, where the same input may produce variation in output. The approach introduces some randomness but offers flexibility in how code is generated.

#### IMPLEMENTATION WITH AI CODE GENERATION

In the Probabilistic approach, both ARS metadata and ADM metadata are used to instruct an AI Code Generator. The AI model also references existing code samples and best practices to refine its output. The generator produces TFL R scripts that ingest ADaM data and produce the final outputs/displays.

The workflow operates through these steps: TFL Designer captures and exports both ARS and ADM metadata, these metadata files instruct the AI Code Generator along with reference codes that help refine the generation, the AI generates TFL scripts, these scripts ingest ADaM data, and finally produce the displays.

This approach differs from the Deterministic approach in that it generates complete TFL code directly, rather than producing intermediate ARD-generation scripts. The AI model can adapt its code generation based on patterns it has learned from reference implementations, potentially offering more flexibility in handling edge cases.

## COMPARATIVE EVALUATION

#### IMPLEMENTATION REQUIREMENTS

The Deterministic approach requires ARS metadata as a foundation, along with well-tested functions or macros for statistical operations. Organizations must also establish internal processes for transforming

ARDs into display formats. This requires upfront investment in building and validating the statistical operation library.

The Probabilistic approach requires both ARS and ADM metadata, reference code demonstrating best practices for statistical operations, and internal processes for human-in-the-loop review and validation. The reference code library serves a different purpose than the Deterministic function library—it provides examples for the AI to learn from rather than executable components to be called.

#### WORKABILITY

The Deterministic approach offers predictable results with the same input always producing the same output. The setup is relatively rigid, requiring predefined functions for all anticipated operations. When issues arise, debugging is straightforward because the logic flow is explicit and traceable.

The Probabilistic approach is prone to variation, with the possibility of different outputs from the same input. However, this approach offers flexible setup, as the AI can potentially handle operations not explicitly programmed. Debugging can be time-consuming because generated code may use unexpected patterns or approaches, requiring careful review to ensure correctness.

#### SUCCESS FACTORS

For the Deterministic approach, success depends heavily on a good understanding of the ARS model, a strong test suite for all functions and macros covering edge cases and ensuring correctness, and well-defined internal standards for code structure and ARD format.

For the Probabilistic approach, success requires thoughtful prompt design that clearly communicates requirements to the AI model, an effective feedback loop where generated code is reviewed and lessons learned are incorporated into future prompts, and high-quality reference code that demonstrates best practices the AI should emulate.

## CONSIDERATIONS FOR IMPLEMENTATION

Using automated code generation with ARS metadata represents a change from traditional TFL generation workflows. Several factors need consideration when implementing either approach.

#### INFRASTRUCTURE REQUIREMENTS

Organizations need appropriate technical infrastructure, such as R Studio workspaces for the R-based approaches discussed here. For the Deterministic approach, this includes package management systems and version control. For the Probabilistic approach, secure access to AI services with appropriate data governance is essential.

#### KNOWLEDGE AND SKILLS

A strong knowledge of ARS metadata and the Logical Data Model is essential for the team members responsible for setting up the metadata. This represents a shift from traditional programming-focused roles to more metadata and configuration-focused work.

#### TEAM COMPOSITION

Team composition may need updating as the balance between statisticians and programmers shifts. Setting up ARS metadata requires more involvement from statistical programming expertise during setup, but since automation is applied, less direct programming is needed during execution. Programmers can become owners and maintainers of the generated scripts rather than writing everything from scratch.

#### CHANGE MANAGEMENT

The use of ARDs in the workflow of TFL generation requires a major shift from the status quo. Traditionally, ADaM datasets are programmed directly into RTF or PDF displays, without an

intermediate step where machine-readable Analysis Results Datasets are created. This change needs to be designed carefully, with the benefits well understood to achieve successful transitions.

A recurring concern from stakeholders is the "black box" problem—passing metadata into a system that produces results without visibility into the process. Both approaches address this differently: the Deterministic approach makes generated code fully visible and editable, while the Probabilistic approach requires robust review processes to validate generated code meets requirements.

## USE CASES AND RECOMMENDATIONS

### When to Use Deterministic Approach

The Deterministic approach is well-suited for organizations that require absolute consistency and reproducibility, have the resources to build and maintain a comprehensive library of statistical functions, work in highly regulated environments where explainability and traceability are paramount, and have standardized analysis patterns that can be codified into reusable components.

Organizations using this approach benefit from predictable behavior and straightforward validation. Once the function library is built and tested, the approach scales well across multiple studies with similar analysis patterns.

### When to Use Probabilistic Approach

The Probabilistic approach may be appropriate for organizations dealing with highly variable or non-standard analyses that are difficult to pre-program, exploring innovative approaches where flexibility outweighs the need for perfect reproducibility, and having robust code review processes in place to validate AI-generated code.

This approach requires strong governance around code review and validation. The ability of AI to generate novel solutions can be valuable, but each output requires careful verification to ensure correctness and appropriateness.

### Hybrid Approaches

Organizations need not choose exclusively between these approaches. A hybrid model might use Deterministic generation for standard, well-understood analyses while employing Probabilistic generation for exploratory or non-standard analyses. This combines the reliability of rule-based generation with the flexibility of AI-assisted generation.

## CONCLUSION

### Impact of Automation

Implementing automated code generation into workflows means a larger focus on metadata setup and a lesser focus on manual programming. Getting the ARS metadata in good shape requires focused effort, likely taking more time than traditional metadata setup. However, this effort is rewarded by decreased manual programming efforts, making use of the automation capabilities provided by ARS and supporting tools.

Time spent on repetitive programming tasks is reduced, freeing resources to spend more time on complex tasks such as configuration of statistical procedures and handling of special cases. This allows for more productive use of resources, with automation handling repetitive work while human expertise focuses on challenging problems.

As a result of automation, the overall cost of programming decreases while quality, consistency, and efficiency increase. This aligns with industry goals of producing more reliable results at less cost, benefiting all stakeholders and ultimately the patient.

### Key Learnings

Both approaches address the critical issue of transparency. Stakeholders are uncomfortable with pure "black box" solutions. The Deterministic approach achieves transparency through visible, editable code. The Probabilistic approach requires transparency through robust review processes and clear documentation of how AI-generated code was validated.

Each organization has unique context and requirements. Automation solutions need flexibility to accommodate different workflows, standards, and preferences. Going forward, successful tools will emphasize configurability, allowing users to provide custom code snippets, define their own operations, and adapt the system to their specific needs.

### FUTURE DIRECTIONS

The next steps for both approaches involve implementations into existing workflows. Real-world usage will reveal areas for improvement and enhancement. For Deterministic approaches, expanding the library of statistical operations and improving metadata interfaces will be key. For Probabilistic approaches, refining prompt engineering and validation workflows will be essential.

The convergence of new standards (ARS), open-source technology (R packages), and AI capabilities creates unprecedented opportunities for TFL automation. By understanding the strengths and limitations of both Deterministic and Probabilistic approaches, organizations can make informed decisions about which approach, or combination of approaches, best serves their needs.

## REFERENCES

[1] CDISC, "Analysis Result Standard," April 2024. [Online]. Available: https://www.cdisc.org/standards/foundational/analysis-results-standard.

[2] CDISC, "Analysis Results Standard User Guide v1.0," April 2024. [Online]. Available: https://wiki.cdisc.org/display/ARSP/Analysis+Results+Standard+User+Guide+v1.0

[3] Clymb Clinical, "TFL Designer," [Online]. Available: https://clymbclinical.com/

[4] CDISC, "ARS V1 Examples," April 2024. [Online]. Available: https://github.com/cdisc-org/analysis-results-standard/tree/main/workfiles/examples/ARS%20v1

## ACKNOWLEDGMENTS

The author would like to acknowledge Bhavin Busa (Co-Founder of Clymb Clinical and Product Owner of CDISC's ARS) for his continued guidance and support on this topic. Furthermore, acknowledgment is extended to Richard Marshall (Principal Architect of ARS model) for his technical insights and assistance with understanding key concepts of the Analysis Results Standard.

## RECOMMENDED READING

eTFL Portal utilizing ARS: https://www.cdisc.org/events/webinar/introducing-new-cdisc-etfl-portal

ARS Public Review: https://www.cdisc.org/events/webinar/analysis-results-standard-public-review

CDISC ARS GitHub Repository: https://cdisc-org.github.io/analysis-results-standard/

## CONTACT INFORMATION

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

Author Name: Malan Bosman

Company: Clymb Clinical

Email: mbosman@clymbclinical.com

Website: https://clymbclinical.com/