Paper SM10

CDISC ARS Template Code: Gearing up with automation in TFL Programming

Anna Yaggi, Clymb Clinical, Burlington, USA

Malan Bosman, Clymb Clinical, Bloemfontein, South Africa

ABSTRACT

With the advent of the CDISC Analysis Results Standard (ARS), TFL automation is being re-imagined. The concept of using structured metadata and generating results in an Analysis Results Dataset (ARD) format first, allows for new ways to automate TFLs. One particularly exciting opportunity for automation using ARS, is utilizing the "Analysis Method Code" concept. In this PHUSE 2025 paper, Richard Marshall referred to it as "Gear 5" of TFL Automation. It allows for macro-like code to be set up for each Analysis Method (e.g. frequency counts, or summary statistics, etc.), along with parameters sourced from the ARS metadata. In an R implementation, the 'cards' and 'cardx' packages (both part of 'pharmaverse' offer uniquely efficient functions for this use case. This presentation will dive into how these functions are used as template codes within the ARS metadata, and how we can create dynamic code to generate reliable, automated results for TFLs.

INTRODUCTION

Statistical programmers around the world write and re-write code daily to produce results in the form of tables from clinical research data. Mostly, it's code to do simple calculations like counting subjects per treatment or summarizing the age of subjects. Sometimes, it's more complex analyses, like p-values or calculating risk difference between treatments. Regardless of the type of analyses and code written, one can't help but wonder: Is there a more efficient way? Is it really necessary for the same programmers in the same company to write the same proven code for the same analyses over and over again? Is there a way to re-use reliable code snippets (call it macros or functions) to reduce the effort required to generate clinical results? And if so, is there a proven industry standard or framework for doing so systematically?

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 therefore provides a Logical Data Model that describes analysis results and associated metadata. 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 one result per row and context for each result, which is known as an Analysis Results Dataset (ARD).

To create a result (and by implication many results to create an ARD), various model components need to be considered in combination. In Figure 1 below, we focus on a handful of key components from the model, and their interaction is as follows: each Analysis contains an Analysis Set (typically the subject population), Data Subsets (conditions describing the selection of data to be included in the analysis), Analysis Grouping (describing how to group data for analysis) and Analysis Method (the set of statistical operations applied to the data).

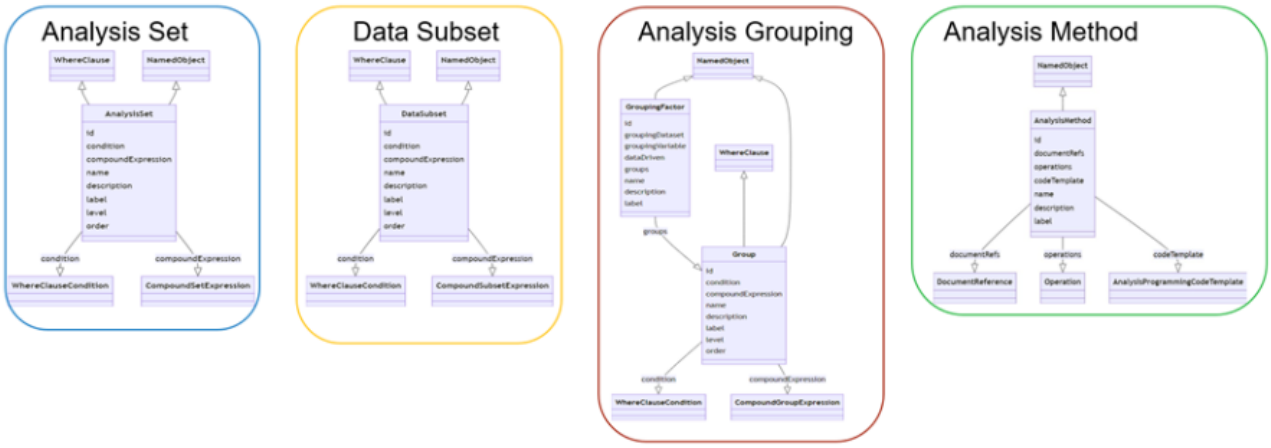

Figure 1 - ARS Logical Model Classes and their Metadata Components

ANALYSIS METHOD CODE TEMPLATE

The "AnalysisMethod" class of the ARS metadata is of particular importance to this discussion. It is described as "a set of one or more statistical operations" and has the attribute "codeTemplate": "template programming statements used to perform the statistical operations for any analysis that uses this method". These template programming statements are where we can provide re-usable parameterized template code (code snippets like macros or functions) to perform calculations for the analysis method. The values populated by these parameters come from various ARS metadata components (such as Analysis Sets, Analysis Groupings, etc.)

EXPLANATORY EXAMPLE

Some of these ARS concepts and names could be new to some. Let's look at a simple example for explaining the concepts:

Characteristics	Xanomeline Med Dose (N=XX) n (%)	Xanomeline High Dose (N=XX) n (%)	Placebo (N=XX) n (%)
Age			
n	XX	XX	XX
Mean (SD)	XX.X (XX.XX)	XX.X (XX.XX)	XX.X (XX.XX)
Median	XX.X	XX.X	XX.X

Figure 2 - - Example analysis: summarizing age

Note the following:

- The "analysis" is described as summarizing the age of subjects per treatment
- The "analysis method" would be the set of statistical operations to be performed, like "n, mean, standard deviation, median"
- The "analysis method code template" is then the code snippet (like a 'proc means' macro in SAS, or a summarizing function in R, for example) used to do this summary calculation for the set of operations. In the next section, we will be looking at a specific system of template code to be used for an implementation.

`CARDS` AND `CARDX` R PACKAGES

When implementing the concept of "analysis method code template" in R and selecting how to construct re-usable code, there are many packages, functions and structures to choose from. For example, one could use dplyr's "group_by" and "summarize" functions to get variable summaries, or a base R approach.

Another option is working with the `cards` and `cardx` packages, designed specifically to wrap other statistical R functions and produce results in a consistent format. This format is known as an analysis results dataset, or ARD (`cards` being an abbreviation of CDISC ARDs, and `cardx` indicating "extra functions"). An ARD is structured as a flat dataset with one result per row, and columns providing context to the result (such as grouping levels, format, and statistic names). An example of such an ARD and the `cards` function call `ard_summary` is shown in figure 3:

```
ard_summary(ADSL, by = "ARM", variables = "AGE")
#> {cards} data frame: 24 x 10
#>    group1 group1_level variable stat_name stat_label   stat
#> 1     ARM      Placebo      AGE         N          N     86
#> 2     ARM      Placebo      AGE      mean       Mean 75.209
#> 3     ARM      Placebo      AGE        sd         SD   8.59
#> 4     ARM      Placebo      AGE    median     Median     76
#> 5     ARM      Placebo      AGE       p25         Q1     69
#> 6     ARM      Placebo      AGE       p75         Q3     82
#> 7     ARM      Placebo      AGE       min        Min     52
#> 8     ARM      Placebo      AGE       max        Max     89
#> 9     ARM    Xanomeli…      AGE         N          N     84
#> 10    ARM    Xanomeli…      AGE      mean       Mean 74.381
#> i 14 more rows
```

Figure 3 - ARD example for summarizing AGE by ARM using `cards` package

As a quick note on the ARDs produced directly from functions in `cards` and `cardx` - there is no explicit link between the ARDs (directly produced by the R functions) and ARS metadata. ARS metadata has clear ID-based links to e.g. Analyses, Methods, DataSubsets, AnalysisGroupings, and other metadata components, providing full traceability to the analysis results. Though these metadata components are not included in `cards` and `cardx` produced ARDs, the required IDs can be merged onto ARDs afterwards as required. An example of such an ARD with ARS-enriched linkage in the last four columns is shown in Figure 4 below:

group1	group1_level	variable	context	stat_name	stat_label	stat	fmt_fun	warning	error	operationid	AnalysisId	MethodId	OutputId
TRT01A	Placebo	AGE	continuous	N	N	315	0	NULL	NULL	Mth_24_01_n	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	mean	Mean	43.79429	1	NULL	NULL	Mth_24_02_Mean	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	sd	SD	8.363728	1	NULL	NULL	Mth_24_03_SD	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	median	Median	44	1	NULL	NULL	Mth_24_04_Median	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	p25	Q1	38.4	1	NULL	NULL	Mth_24_05_Q1	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	p75	Q3	49.6	1	NULL	NULL	Mth_24_06_Q3	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	min	Min	14.4	1	NULL	NULL	Mth_24_07_Min	An_06	Mth_24	Out_01
TRT01A	Placebo	AGE	continuous	max	Max	64	1	NULL	NULL	Mth_24_08_Max	An_06	Mth_24	Out_01

Figure 4 - an ARD produced by the `cards` package, with ARS-linked metadata in the last four columns for traceability

The `cards` and `cardx` packages are widely adopted across the industry (>50 000 downloads for `cards` and > 44 000 downloads for `cardx` per month as at the time of writing). The packages are

also structurally linked to the `gtsummary` R package (>52 000 downloads per month). In summary, the strength of `cards` and `cardx` packages for being used as template code are as follows:

- Consistent, concise function call style across a variety of statistical calculations
- Consistent result format as an ARD (which aids structuring results)
- Incorporating / wrapping reputable R packages
- Wide adoption across the clinical research industry

For the above reasons, making use of `cards` and `cardx` in building template code is a solid choice for teams working in R and looking for ways to re-use code for statistical analyses.

IMPLEMENTATION

Let's examine how cards functions are incorporated into Analysis Method Code Templates. Consider a template for generating summary statistics on a continuous variable:

```
Analysis_ARD <- ard_summary(data = filtered_data,

                            by = c(byvariables_here),

                            variables = analysisvariable_here )
```

In this template, `byvariables_here` and `analysisvariable_here` are placeholders. When generating code for a specific analysis (e.g., AGE by TREATMENT), these would be replaced with actual variable names:

```
Analysis_ARD <- cards::ard_summary(data = filtered_ADSL,

                                   by = c(TRT01A),

                                   variables = AGE)
```

The same template could be reused for weight, height, or any other continuous variable - only the `analysisvariable_here` parameter would change. This represents a significant efficiency gain: one template code serves many analyses.

Using the same concept as above, more complex analysis methods can be covered with template code. An example from a `cardx` package could be Chi-Square p-value:

```
df3_analysisidhere <-

    cardx::ard_stats_chisq_test(by = groupvar1here,

                                data = df2_analysisidhere,

                                variables = groupvar2here)|>

    dplyr::filter(stat_name == "p.value") |>

     dplyr::mutate(operationid = dplyr::case_when(

                                  stat_name == 'p.value' ~ opid1here'))
```

As a final example, parameterized template code for Fisher's exact test is provided. In this example, it is shown that pre-processing steps for the statistical analysis step (`cards` / `cardx` function) can also be utilized:

```
full = df_pop |>

     dplyr::filter(armfilterhere)


success = df2_analysisidhere |>

    dplyr::mutate(FL = 1)


ana = full |>

    dplyr::left_join(success |>

                       dplyr::select(anavarhere,

                                     FL),

                     by = "anavarhere") |>

    dplyr::mutate(FL = ifelse(is.na(FL), 0, FL))


df3_analysisidhere = ana |>

  cardx::ard_stats_fisher_test(by = groupvar1here,

                                       variables = FL) |>

   dplyr::filter(stat_name %in% c('estimate',

                                  'conf.low',

                                  'conf.high')) |>

   dplyr::mutate(operationid = dplyr::case_when(

                                    stat_name == 'estimate' ~ 'opid1here',

                                    stat_name == 'conf.low' ~ 'opid2here',

                                    stat_name == 'conf.low' ~ 'opid3here'))
```

PARAMETERS

To facilitate parameter substitution, implementation tools may define "constructs" - predefined objects that can be referenced in templates and automatically populated from ARS metadata. These constructs can be simple, or complex:

Simple Variable Constructs

Direct references to single metadata elements, such as dataset names, analysis variables, or grouping variables. For example:

- `analysis_variable` - The variable being analyzed

Complex Constructs

Dynamic objects built from multiple metadata pieces. These might include:

- `trt_filter_stm` - A constructed filter statement defining a subset of treatments

These constructs are populated for each analysis, and are used as parameters to the template code. They help the template code turn into usable code for the analysis.

MAKING USE OF SIERA R PACKAGE

The `siera` R package represents and implementation where ARS metadata is ingested by an R function (`readARS`), and meta-programmes R scripts which produce ARDs (one ARD-creating script per output in the ARS metadata file). It serves as a run-time engine for replacing the parameterized `cards`-based Analysis Method Code Template code with the actual ARS components represented by the parameters in the Analysis Method Code Parameters.

CONCLUSION

The convergence of the CDISC Analysis Results Standard, Analysis Method Code Templates, and the cards/cardx R packages represent a powerful approach to TFL automation. By embedding cards functions within parameterized templates and driving code generation from rich ARS metadata, we can achieve new levels of efficiency, quality, and consistency.

This "Gear 5" approach to automation delivers tangible benefits: dramatic reductions in programming time, improved quality through standardization, enhanced maintainability through centralized templates, clear audit trails from specification to result, and streamlined validation processes. These benefits translate directly to faster study timelines, reduced costs, and higher quality submissions.

Successful implementation requires investment in infrastructure, processes, and capabilities. Organizations must develop expertise in ARS metadata, build libraries of validated templates, establish workflows that integrate with existing processes, and foster collaboration between biostatisticians, programmers, and other stakeholders.

The cards and cardx packages are purpose-built for this use case, with consistent interfaces, standardized ARD outputs, built-in metadata capture, and flexible parameterization. Their integration into the pharmaverse ecosystem ensures they will continue to evolve and improve alongside other open-source clinical development tools.

As the industry continues its journey toward comprehensive TFL automation, the combination of ARS, template-based code generation, and specialized tools like cards/cardx will play an increasingly central role. Organizations that invest in these capabilities now will be well-positioned to reap the benefits of improved efficiency, quality, and innovation in clinical programming.

REFERENCES

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

[2] Marshall, R., "Shifting geARS: Levels of Code-Driven Automation in the Analysis Results Standard," PHUSE US Connect 2025.

[3] cards package documentation. [Online]. Available: https://insightsengineering.github.io/cards/

[4] cardx package documentation. [Online]. Available: https://insightsengineering.github.io/cardx/

[5] CDISC, "ARS User Guide," Version 1.0, 2024. [Online]. Available: https://www.cdisc.org/standards/foundational/analysis-results-standard

[6] pharmaverse. [Online]. Available: https://pharmaverse.org/

[7] Bosman, M., "Advancing TFL Automation: R-Based Meta-Programming Powered by ARS," PHUSE Connect 2020.

ACKNOWLEDGMENTS

We 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 project. Furthermore, we would like to acknowledge Richard Marshall (Principal Architecture of ARS model) for his technical insights and assistance with understanding key concepts.

CONTACT INFORMATION

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

Author Name: Anna Yaggi

Company: Clymb Clinical

Email: ayaggi@clymbclinical.com

Website: https://clymbclinical.com/

Author Name: Malan Bosman

Company: Clymb Clinical

Email: mbosman@clymbclinical.com

Website: https://clymbclinical.com/