Paper SM13

Let the Machine Do the Coding: A SAS-based Framework for Automating TFL Generation with ARS Metadata

Julie Wachara, Clymb Clinical, Burlington, MA, United States

ABSTRACT

In today's fast-paced clinical trial environment, efficiency and adaptability are key to meeting programming timelines. This paper presents an approach to automating study programming tasks by leveraging TFL shells and CDISC ARS metadata. By automating much of the programming, this framework reduces manual effort and ensures consistency across deliverables. This SAS®-based solution generates code templates for all required TFLs, pre-populating key sections with accurate, study-specific information. Its metadata-driven automation accommodates frequent TFL additions and modifications, offering flexibility during early study stages. For project managers and lead programmers, this approach optimizes oversight, boosts productivity, and positions automation at the core of the programming process. Let the machine do the programming for you.

INTRODUCTION

Clinical Study Reports are composed of several Tables, Listings and Figures which are often updated and modified throughout the study period. Organizing and setting up the study programming environment to create these outputs and accommodate new requests is key to maintaining timely reviews and submissions. Current approaches to automation are effective, however, these can be challenging to implement as they often utilize controlled repositories or involve back-end processing that may not be possible or accessible to most programming teams.

In a traditional stand-alone workflow, programmers typically use a combination of macros and programs from previous studies to set up the programming environment. While it is helpful to get started, updating individual study level TFL programs to match the current study, especially when there are hundreds of outputs, is time-consuming, repetitive, error-prone, and offers poor traceability.

This paper provides a SAS-based metadata-driven approach to automate this process in three steps:

- Building mock shells using CDISC ARS Standard as a metadata repository.
- Using the extracted metadata to apply statistical operations using SAS-based macros.
- Generating executable programs for each TFL.

ANALYSIS RESULTS STANDARD

Background

The CDISC Analysis Results Standard (ARS), first published in April 2024, provides a logical data model that defines analysis results along with their associated metadata. The ARS model provides a structured framework to hold specification of the input datasets, derivations, and statistical tests used to produce the results represented on TFLs.

This not only improves traceability and consistency across submissions but also provides standardized logical descriptions of analyses and the metadata used to produce the results.

Structure

In the ARS Logical model a reporting event, for example a CSR, is composed of multiple outputs (TFLs) each with several analyses. Each analysis contains an Analysis Set, the subject population, Data Subset, conditions describing the selection of data to be included in the analysis, Analysis Grouping, characteristics used to cluster the subject populations, and Analysis Method, the statistical operation applied to the data. The metadata for the reporting event is organized into classes with fields that house the metadata elements as shown in Figure 1.

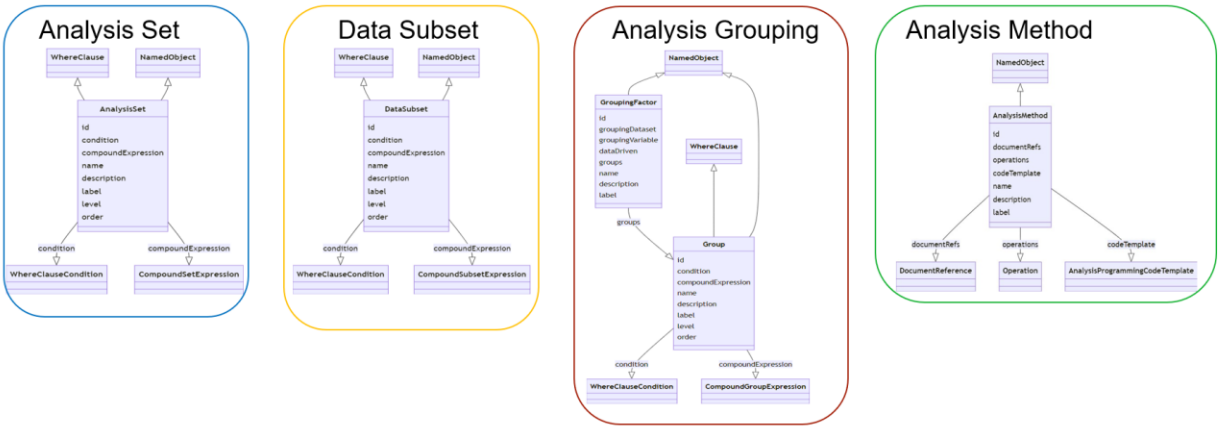

Figure 1. ARS Logical Model Classes and their Metadata Components

For this framework we will be focusing on classes that define the expected outputs, their grouped analyses and the statistical operations performed to get the results. These include:

- Output
- Analysisoutputcategorization
- Analysisset
- Analysis
- Group
- Groupingfactor
- Orderedgroupingfactor
- Analysisgroupings
- Operation
- OrderedlistItem
- Displaysubsection

Detailed descriptions of each class, its fields, and how it is related to other classes can be found in the CDISC ARS Model Documentation [1] and accompanying user guide [2].

APPLICATION FRAMEWORK

In traditional programming, the CDISC Analysis Results Metadata (ARM) Define.xml is typically generated retrospectively, after TFL programming. However, a metadata-driven approach shifts the focus to prospectively centralizing the programming workflow around a standardized TFL specification. By leveraging the study Protocol, Statistical Analysis Plan (SAP), and CDISC standard ADaM library, programmers can manually add ARS-compliant metadata to an Excel sheet or use an automation tool. For this paper, we used the community version of TFL Designer [3] which enables programmers to integrate CDISC ARS metadata into TFL shells. The granularity of metadata within the shell determines the available metadata for downstream use, allowing flexibility - whether minimal or exhaustive - based on specific requirements. Figure 2 below shows the metadata-driven TFL deliverable workflow we are using and introduces ***atlas***.

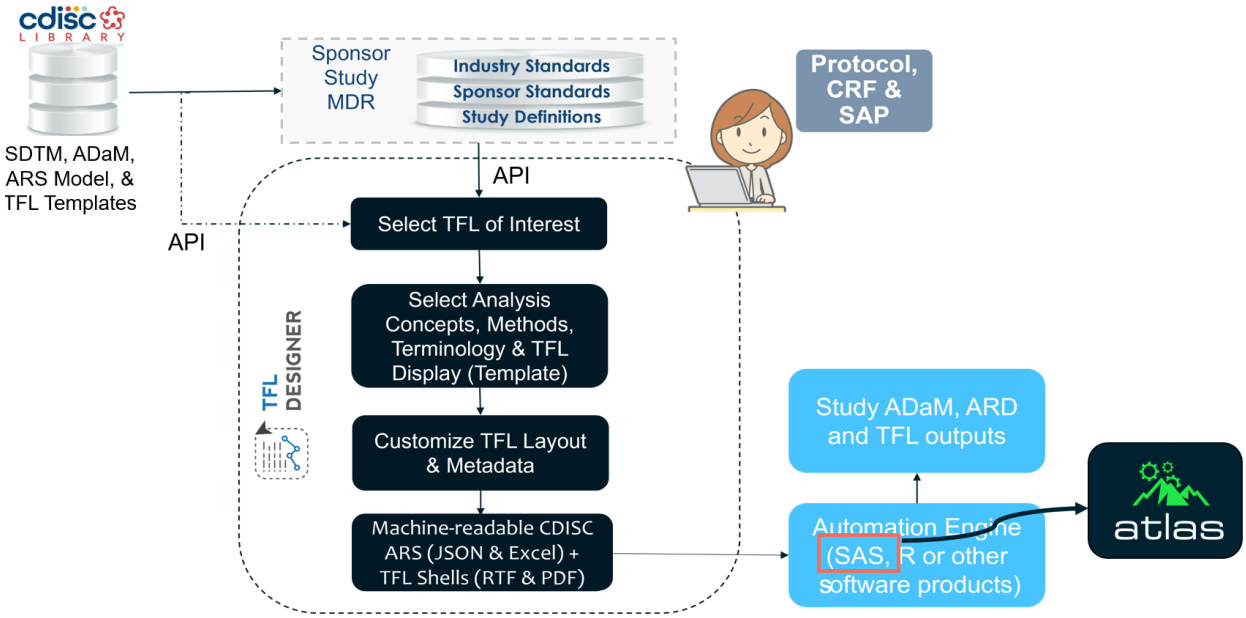

Figure 2: Metadata-driven TFL deliverable workflow

atlas is a SAS-based automation engine developed by our team that ingests CDISC ARS metadata, applies statistical operations, and generates ready-to-use, executable SAS programs for each TFL. Figure 3 below shows how *atlas* is incorporated into our TFL deliverable workflow. This paper outlines how *atlas* uses the metadata exported from the mock shells to select macros from the SAS macro library and create TFL programs with the macro populated with the TFL-specific variables.

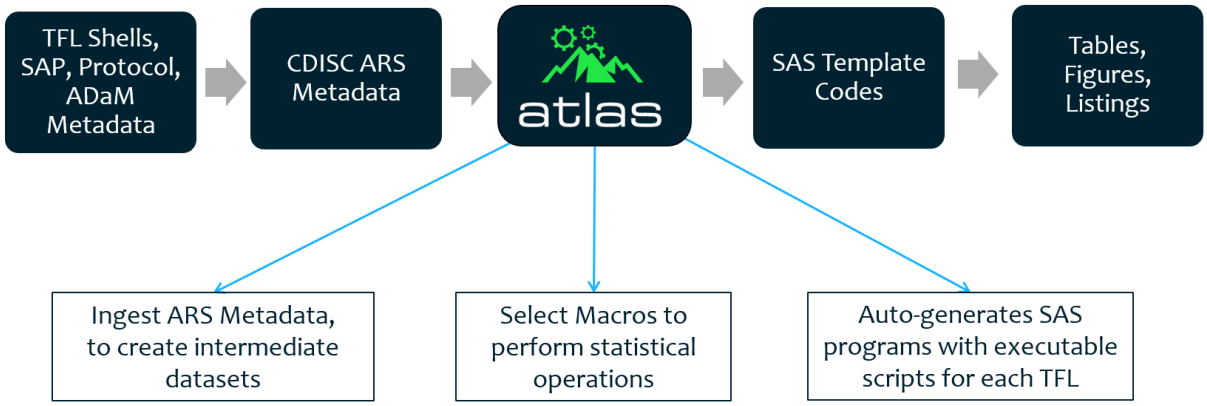

Figure 3: ***atlas*** Meta-programming Process Flow: Shells to TFLs

Entry and Export of Metadata

Programmers use the protocol, SAP and ADaM metadata to populate the Metadata in the TFL Shells using the CDISC ARS Standard Library as shown in Figures 4 and 5 below. This metadata is then exported from the automation tool and filtered for the downstream processing. The mock shell templates used for this example were retrieved from the eTFL Portal [5], a resource provided by CIDISC with example CDISC ARS compliant packages.

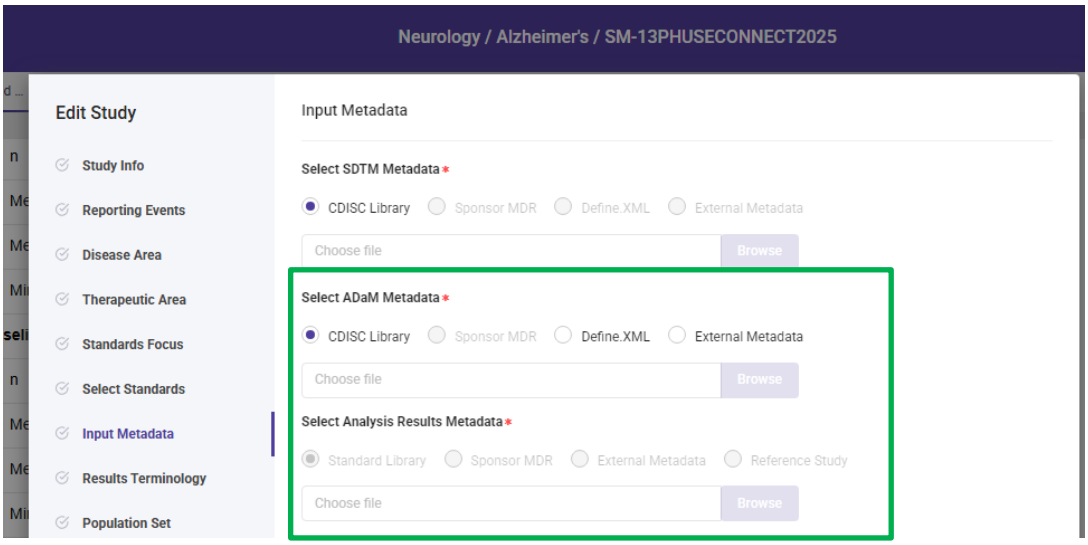

Figure 4. Example of Study Set-up Including Metadata from CDISC ADaM and ARS Standard Library.

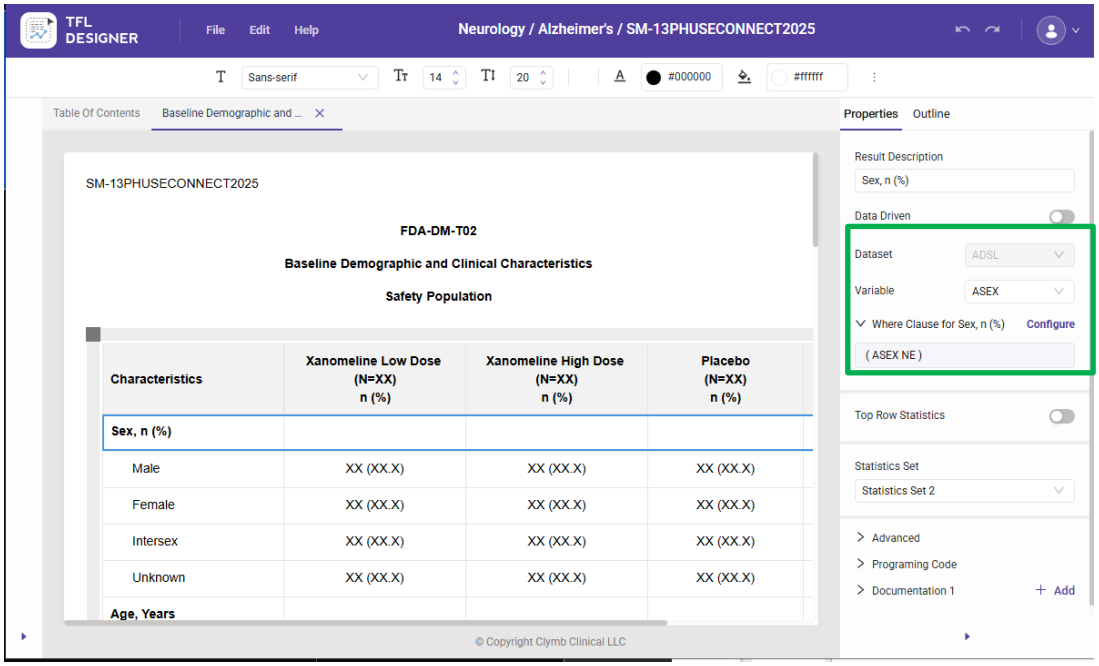

Figure 5. Example of Metadata Input on Mock Shell using CDISC ARS Standard Library.

META-PROGRAMMING

Organize Metadata

Once the shells are populated, the ARS metadata can be exported in JSON or excel format. This paper utilized the JSON export which was then converted to SAS datasets following the ARS Class structure. The code used for this conversion is available on CDISC ARS repository on GitHub under SAS utilities [4].

The goal in this step is to keep the key fields (columns) from the class datasets that contain the metadata necessary to identify the analyses category and operations performed on each TFL. This intermediate dataset can then be parsed into defined macro variables and used to dynamically call the macro based on the statistical operation. Figure 6 shows an example of the relationship between the class datasets and how the intermediate dataset is constructed from the metadata.

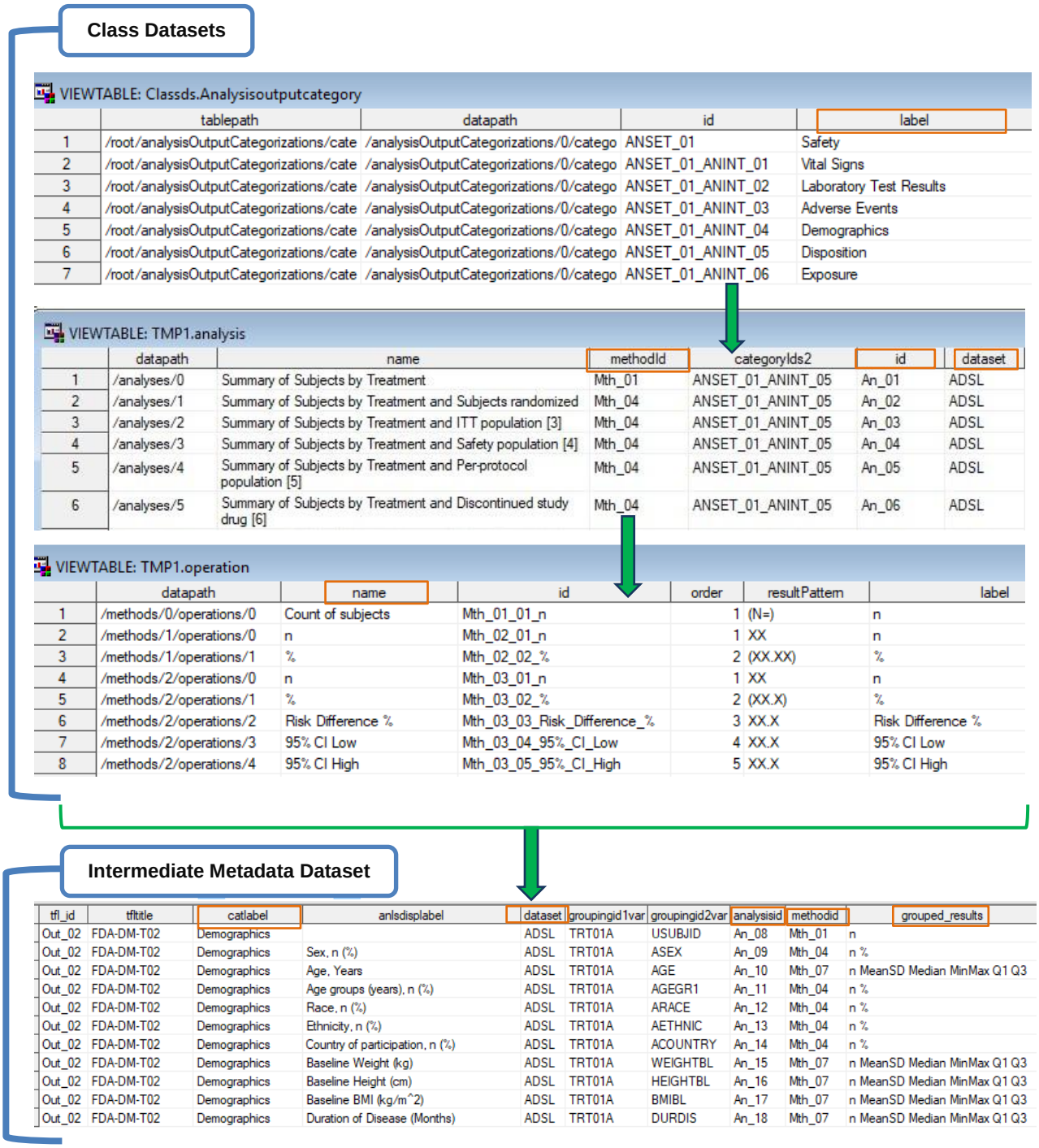

Figure 6. Relationship between class datasets and resulting intermediate metadata dataset.

Several factors were considered to determine which metadata should be kept including the structure of the ADaM datasets referenced and the level of macro specificity and utility. The SAS macros used for this automation were a combination of TFL template-specific and global macros. The TFL template-specific macros focused on the TFL analysis category and used the analysis method and analysis category in the metadata as a key to identify the macro used and therefore which template to group the analyses. The global macros focused on general information such as global headers, TFL names, titles, and footnotes.

Assign SAS Macros

The second step in this automation is the assignment of SAS Macros by statistical operation. This was done by grouping the TFLs by analysis category and assigning each operation to the appropriate macros within the SAS macro library. For example, macro eventfreq below in the macro library is used to calculate frequency and requires 7 macro variables to perform this operation: inds, acrossvar, freqvar, freqvarfmt, order, rowlabel, and fmtlib.

```
/* Calculating frequency of variable by across variable */
%eventfreq(inds=<Enter Input Dataset Name>,
           acrossvar=<Enter Across Variable Name>,
           freqvar=<Enter Frequency Variable Name>,
           freqvarfmt=<Enter Format Associated w/ Freqvar>,
           order=<Assign Order Number>,
           rowlabel=%bquote(<Enter Row to Appear Before the Variable Summary>),
           fmtlib=<Enter Format Library Name>);
```

In Figure 7 methodid Mth_04 is associated with the frequency statistical operation (n %) in the intermediate dataset. The metadata variables on each row associated with this method are then parsed into the macro variables and grouped by analysis order (column analysisid) and ouput (column tfl_id).

tfl_id	tfltitle	catlabel	anlsdisplabel	dataset	groupingid1var	groupingid2var	analysisid	methodid	grouped_results
Out_02	FDA-DM-T02	Demographics		ADSL	TRT01A	USUBJID	An_08	Mth_01	n
Out_02	FDA-DM-T02	Demographics	Sex, n (%)	ADSL	TRT01A	ASEX	An_09	Mth_04	n %
Out_02	FDA-DM-T02	Demographics	Age, Years	ADSL	TRT01A	AGE	An_10	Mth_07	n MeanSD Median MinMax Q1 Q3
Out_02	FDA-DM-T02	Demographics	Age groups (years), n (%)	ADSL	TRT01A	AGEGR1	An_11	Mth_04	n %
Out_02	FDA-DM-T02	Demographics	Race, n (%)	ADSL	TRT01A	ARACE	An_12	Mth_04	n %
Out_02	FDA-DM-T02	Demographics	Ethnicity, n (%)	ADSL	TRT01A	AETHNIC	An_13	Mth_04	n %
Out_02	FDA-DM-T02	Demographics	Country of participation, n (%)	ADSL	TRT01A	ACOUNTRY	An_14	Mth_04	n %
Out_02	FDA-DM-T02	Demographics	Baseline Weight (kg)	ADSL	TRT01A	WEIGHTBL	An_15	Mth_07	n MeanSD Median MinMax Q1 Q3
Out_02	FDA-DM-T02	Demographics	Baseline Height (cm)	ADSL	TRT01A	HEIGHTBL	An_16	Mth_07	n MeanSD Median MinMax Q1 Q3
Out_02	FDA-DM-T02	Demographics	Baseline BMI (kg/m^2)	ADSL	TRT01A	BMIBL	An_17	Mth_07	n MeanSD Median MinMax Q1 Q3
Out_02	FDA-DM-T02	Demographics	Duration of Disease (Months)	ADSL	TRT01A	DURDIS	An_18	Mth_07	n MeanSD Median MinMax Q1 Q3

```
proc sql ;
  select count (mthdord) into :mthdord from analyses&j;
quit;

    %do k=1 %to &mthdord.;
       data mthdord&k.;
            set analyses&j;
            where mthdord = &k.;
            if catlabel = "Demographics" then do;
                if methodId = "Mth_04" then do;
                    call symputx(cats('anlsnam', &k.), analysisname);
                    call symputx(cats('dsn', &k.), dsn );
                    call symputx(cats('grpvar1', &k.),grpvar1);
                    call symputx(cats('grpvar2', &k.), grpvar2);
                    call symputx(cats('resfmt', &k.), resfmt);
                    call symputx(cats('anlsord', &k.), anlsord);
                    call symputx(cats('library', &k.), library);
                    call symputx(cats('anlslab', &k.), anlsdisplabel);
                end;
                else if methodId = "Mth_07" then do;
                    call symputx(cats('anlsnam', &k.), analysisname);
                    call symputx(cats('dsn', &k.), dsn );
                    call symputx(cats('grpvar1', &k.),grpvar1);
                    call symputx(cats('grpvar2', &k.), grpvar2);
                    call symputx(cats('grpreslt', &k.), grouped_results);
                    call symputx(cats('nmaxdigt', &k.), nmaxdigt);
                    call symputx(cats('vmaxdigt', &k.), vmaxdigt);
                    call symputx(cats('n_dec', &k.), n_dec);
                    call symputx(cats('anlsord', &k.), anlsord);
                    call symputx(cats('anlslab', &k.), anlsdisplabel);
                end;
            end;
        run;

    %end;
```

Figure 7. Example of Macro assigned by analysis category and analysis method.

Generation of Executable Code

The third step dynamically creates ready-to-use executable SAS programs for individual TFL programs with codes. Figure 8 shows an example of the template code used to generate the program for the demographics table FDA-DM-T02 with the eventfreq macro populated for each analysis row that had Methodid Mth_04. In addition to the analysis macros, the program header, study-specific set up files, and titles and footnotes were also added to the program file. The result is the creation of a SAS file for each TFL containing executable script. Figure 9 shows the code in the generated program for the demographics table FDA-DM-T02. These TFLs programs can then be batch run to produce the datasets for review or validation.

```
/* Step 3: Dynamically generate the TFL SAS program with executable scripts */
 filename prog&j "File location\&tfl_type._&pgmtitle._&tfl_cat..sas";
 data _null_;

    file prog&j;
    *Add header properties;
    put "*****************************************************************";
    put "* PROGRAM:                    &tfl_type._&pgmtitle._&tfl_cat..sas";
    put "* DATE:                       &currdate.";
    put "* AUTHOR                      ";
    put "* INPUT PATH and data:        F:\TFL Designer\TFL-Automation\data\adam\sm-13\adam.&tfl_dsn.";
    put "* INPUT PATH and data:        File location\adam.&tfl_dsn.";
    put "* OUTPUT PATH and output name: F:\TFL Designer\TFL-Automation\outputs\tables\primary
                                       &tfl_type._&pgmtitle._&tfl_cat..rtf,";
    put "*                             &tfl_type._&pgmtitle._&tfl_cat..sas7bdat";

    *add other study level maros and libraries as needed;
    put "%include 'F:\TFL Designer\TFL-Automation\tools\sm13_setup.sas';";
    put " ";
    put "%nrstr(%let) tfldsn = &tfl_type._&pgmtitle._&tfl_cat. ;";
    put " ";

    *Assign macro based on methodid and analysis category;
            %if &Mth04_min. > 0 %then %do m = &Mth04_min. %to &Mth04_max.;
                put " ";
                put "***&&anlsnam&m.***;";
                put "%nrstr(%eventfreq)(inds=&&dsn&m., ";
                put "           acrossvar= &&grpvar1&m., ";
                put "           freqvar= &&grpvar2&m., ";
                put "           freqvarfmt= &&resfmt&m., ";
                put "           order= &&anlsord&m., ";
                put "           rowlabel=%nrstr(%bquote)(&&anlslab&m.), ";
                put "           fmtlib=&&library&m.";
                put "           );";
            %end;

            %if &Mth04_min. > 1 or &Mth07_min. > 1 %then %do;
                put "%nrstr(%stack)(outds=%nrstr(&tfldsn.),";
                put "       varlist=&&anlsvar&j.);";
            %end;


               *add code to output TFL dataset ;
               put " ";
               put "data tabdat.%nrstr(&tfldsn.);";
               put "   set %nrstr(&tfldsn.);";
               put "   keep order column var: total;";
               put "run; ";

               *add global macros for titles and footnotes;
               put " ";
               put "%nrstr(%titles_footnotes)(tfltype=&tfl_type., tflnum=&tfl_id.);";
               put " ";
               run;
```

Figure 8. Example of Template code to create table program with assigned macro variables.

```
************************************************************
* PROGRAM:                    t_FDA_DM_T02_demo.sas
* DATE:                       07FEB2025
* AUTHOR
* INPUT PATH and data:        F:\TFL Designer\TFL-Automation\data\adam\sm-13\adam.ADSL
* OUTPUT PATH and output name: t_FDA_DM_T02_demo.rtf, t_FDA_DM_T02_demo.sas7bdat
* ASSUMPTION:
* ------------------------------------------------------------
* CHANGE HISTORY:
* Date        Author      Code Change History Description
* ------------------------------------------------------------
* yyyymmdd    A. Prog     <Code Change description/reason 1>
************************************************************;
proc datasets library=work kill nolist;
quit;
```

```
%include 'F:\TFL Designer\TFL-Automation\tools\sml3_setup.sas';

%let tfldsn = t_FDA_DM_T02_demo ;


***Summary of Subjects by Treatment and Sex, n (%)***;
%eventfreq(inds=adam.ADSL,
           acrossvar= TRT01A,
           freqvar= ASEX,
           freqvarfmt= NONE,
           order= 2,
           rowlabel=%bquote(Sex, n (%)),
           fmtlib=work
           );

***Summary of Subjects by Treatment and Age groups (years), n (%)***;
%eventfreq(inds=adam.ADSL,
           acrossvar= TRT01A,
           freqvar= AGEGR1,
           freqvarfmt= NONE,
           order= 4,
           rowlabel=%bquote(Age groups (years), n (%)),
           fmtlib=work
           );

 ***Summary of Subjects by Treatment and Duration of Disease (Months)***;
 %univars(inds=adam.ADSL,
            acrossvar= TRT01A,
            univars= DURDIS,
            stats= summstat,
            ndig= 2,
            ndvar= 3,
            ndecm=0,
            order= 11,
            rowlabel=%bquote(Duration of Disease (Months)),
            );
 %stack(outds=&tfldsn.,
        varlist=ASEX AGE AGEGR1 ARACE AETHNIC ACOUNTRY WEIGHTBL HEIGHTBL BMIBL DURDIS);
```

```
data tabdat.&tfldsn.;
    set &tfldsn.;
    keep column var: total;
run;
```

```
%title_footnote(tfltype=T, tflnum=2);
```

Figure 9. Example of Table program with SAS macro populated with metadata.

IMPACT, LEARNINGS and NEXT STEPS

The most significant impact of adopting this framework is the increase in quality and efficiency of the programming pipeline. *atlas* reduces manual programming, eliminates the need for repetitive tasks, and is easily adaptable to study changes.

Adopting this framework, however, also brings along some challenges and topics that require further exploration and development. For instance, training and initial setup of the automation process takes time to create and implement. Additionally, multiple aspects of the programming workflow may have to change including additional ADaM variables

for more robust metadata, updating macros to make them more dynamic, and redefining the validation process to include TFL specification review prior to TFL programming.

Looking ahead, the potential for further development and scalability of this framework is promising. As the library of macros and templates grows, automation can expand to support more complex analyses. Since the macros and templates are not study specific, they can also be reused and applied across multiple studies. a metadata-driven approach also presents a method to detect changes from one specification and/or TFL version to another allowing programmers to keep track of changes to programs as the mock shell is updated. In addition to TFL development, ARS metadata could also be used to generate study trackers that contain the expected outputs specified in the metadata, and their associated datasets. This would reduce the manual creation and updating of the programming task tracker.

CONCLUSION

This SAS-based framework "*atlas*" significantly enhances the efficiency and consistency of study programming tasks by leveraging TFL shells and CDISC ARS metadata. It reduces manual effort, ensures uniformity across deliverables, and accommodates frequent TFL additions and modifications, in turn boosting the team's productivity.

While the initial setup time and the need for additional ADaM variables pose challenges, the benefits far outweigh these obstacles. The framework's ability to generate several metadata-driven TFL programs ready to execute in one run positions automation at the core of the programming process, making the workflow much more efficient.

Overall, the adoption of this framework represents a significant step towards a more streamlined and efficient programming process, ultimately contributing to the advancement of automation processes and letting the machine do the coding!

REFERENCES

1. CDISC ARS Model: https://cdisc-org.github.io/analysis-results-standard/
2. CDISC ARS User Guide version 1.0: https://wiki.cdisc.org/display/ARSP/Analysis+Results+Standard+User+Guide+v1.0
3. TFL Designer Community v1.4.0, 2024: https://tfldesigner.org/
4. CDISC ARS Repository https://github.com/cdisc-org/analysis-results-standard/tree/main/utilities/sas
5. eTFL Portal: https://cdisc.org/kb/etfl

RECOMMENDED READING

- CDISC 360 White Paper, 2021: https://www.cdisc.org/sites/default/files/2021-06/CDISC_360_Project_White_Paper.pdf

CONTACT INFORMATION

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

Julie Wachara
Senior Statistical Programmer
Clymb Clinical
jwachara@clymbclinical.com