PharmaSUG 2020 - Paper AD-084

Metadata-driven Modular Macro Design for SDTM and ADaM

Ellen Lin, Aditya Tella, Yeshashwini Chenna, and Michiel Hagendoorn, Seattle Genetics, Inc.

ABSTRACT

In clinical trial data analyses, macros are often used in a modular or building-block fashion to standardize common data transformations in SDTM or derivations in ADaM to heighten efficiency and quality across studies. A traditional design of such macros is the parameter-driven approach, which provides users with many parameters to control differences on input, output, and data processing from study to study. There are limitations with this kind of design, for example the macro may stop working when unexpected differences arise beyond the scope controlled by parameters, or a lengthy revision and documentation cycle is needed to rework parameter-driven code once more variations are introduced by new studies.

Metadata-driven programming is a much more dynamic approach for macro design, especially when targeting areas where differences between studies are less predictable. This design allows key portions of logic and processing to be driven by study-specific metadata maintained outside the macro instead of being exclusively controlled by parameters. It also greatly simplifies user input through parameters and opens the door for a robust and stable macro library at department level.

This paper will describe the metadata-driven approach in detail, discuss important considerations on design of such metadata to allow sufficient flexibility, and explain several optimal programming techniques for it. We will also provide real-world data processing examples where a traditional parameter-driven macro would be challenging, and the metadata-driven approach fits much better.

INTRODUCTION

In clinical trial data analyses, an impactful piece of the statistical programming infrastructure is to standardize common SDTM data mapping and ADaM derivations as departmental modular macros to reduce custom programming at the study level for improved efficiency and better quality. As industry and company data standards are developed and evolving, more departmental macros become feasible. Meanwhile, significant differences between clinical trial studies still have to exist due to different trial designs, data collection, data analyses, and programming standards. Hence, a departmental macro not only processes standard data algorithms, but also is designed to handle potential differences between studies.

Traditionally, we use macro parameters to control the study-specific portion of the requirements by users. Macro parameters are considered very effective and efficient with simple and predictable differences. When study-specific requirements get more complex, unpredictable, or larger in scope, using parameter control exclusively becomes less user-friendly or even not feasible, which limits the development of the departmental macro library. To expand the programming standardization at department level, we implemented the so-called metadata-driven ("MD" in the rest of this paper) design on some significant pieces of SDTM and ADaM programming. Departmental macros become possible and more useful for all studies by using this approach to control a significant number of study-specific requirements while dealing with the standardized portion. Below are a couple of examples from our MD macro library.

- **Deriving the SDTM standard variable RFPENDTC.** The definition of RFPENDTC (Subject Participation End Date) is standard across all studies, as the latest subject participation date (e.g., assessment, event, or intervention date) based on all data collected in a given clinical trial. End-to-end programming of RFPENDTC can be standardized as a departmental macro only if the differences in identifying data records and date variables from each of the raw data sets can be controlled by users.
- **Raw data cutoff processing.** Raw data cutoff as of a target date is often implemented in clinical trial interim analyses to obtain a static and 'clean' set of data. This kind of data cutoff process is usually based on a set of defined criteria, which could be study and data set-specific. End-to-end programming to process the entire clinical trial database can be standardized as a departmental macro only if the very complex differences can be effectively controlled by users.

MD design is a more dynamic approach compared to an exclusive parameter-driven macro and much needed for the complex data scenarios like the two examples above. In the later sections, we will discuss the MD concept as well as key considerations on related metadata and programming techniques. We will use the raw data cutoff example to further illustrate the design concepts in detail.

WHAT ARE METADATA-DRIVEN MACROS

An MD macro design is a hybrid approach where those predictable and simple differences between studies are controlled by users through macro parameters while complex or unpredictable differences are input through study-specific specifications outside the macro. It is critical to distinguish these two types of differences through the analysis of business requirements. With the implementation of MD programming, a key portion of the macro code is driven dynamically by those study-specific specifications.

Not all data processing macros are good candidates for an MD design. A useful application of the technique is for candidates like the RFPENDTC derivation and raw data cutoff processing where the macros process a large number of data sets and the individual data set names, expected variable or data values within each data set, and / or data set-specific algorithms would need to be specified from study to study.

There are advantages of using the MD approach when applicable. It is a more dynamic programming technique that helps expand programming standardization at the department level by incorporating a large scope of study-specific input into the macro programming. Programming specifications, especially complex ones, may go through frequent changes over time. Separating specifications from the programming code avoids constant changes in programs and reduces error-prone coding; the process also enforces study teams to focus on accurate metadata in the early stage. The design significantly simplifies the macro parameters and provides more user-friendly interfaces.

CONSIDERATIONS ON METADATA

Metadata is ‘data about data’, i.e., programming specifications that describe not only the data characteristics but also the process to support the creation of data and related outputs. In MD design, metadata is considered a ‘living document’ that supports specific portions of macro logic and coding; it needs to be transformed from the traditional ‘non-operational’ formats (free-text style) into structured and executable data to automatically drive the macro coding. Since modular macros are usually developed and tested separately, it is efficient to design and store the supporting metadata separately as well. Below are a few considerations with regard to selecting metadata tools and designing metadata structures:

- **Use Microsoft Excel worksheets for metadata storage and maintenance**. A common approach to structure metadata for a given macro is to store it in one or multiple separate data tables; in most cases, only one data table is needed. Given the small scope of the metadata storage for any given macro, we chose Excel for the creation, maintenance and review of the metadata. Data stored in an Excel sheet with rows and columns can be easily accessed by programming. Users may also take advantage of basic Excel functions to manage the data, e.g., use the sheet protection function to write-protect the metadata to avoid accidental or unauthorized changes.
- **Automate metadata creation as much as possible.** In general, metadata creation and maintenance are a manual process and often require tedious manual data entry. That said, if a significant portion of the data entry is the same or similar across studies, at least partial automation of the metadata process can be implemented to increase efficiency. In our examples, we use a macro to automatically initiate the metadata Excel sheet with all standard column headers and raw data set names populated. This automation step helps reduce manual data entry and minimizes the risk of altering standard structures through manual processing.
- **Capture only the complex study-specific specifications outside of macros**. For example, in the data cutoff processing, specifying individual data set names, how to use data records and variables from each data set, and data set-specific cutoff criteria is not only study-specific but also complex; such information is better stored outside of the macro. On the other hand, the data cutoff date is a single data value that, although analysis specific, can be controlled by users using a parameter; using

an MD approach for controlling this cutoff date may unnecessarily complicate the macro programming.

- **Use standardized structure and formats**. Specifically, consider the following across studies:
    1. Excel sheet column header names must be the same in order to standardize the metadata structure.
    2. A given column must contain a same type of specifications in all rows. For example, if a "DataSet" column is used to specify individual data sets to be processed, that column would contain one data set per row. If the decision is to use two-level data set naming (libname and data set name), then that should be applied consistently to all rows.
    3. Although each macro is associated with a separate metadata worksheet, it is efficient to design column headers consistently across macros when applicable. E.g., both RFPENDTC and the raw data cutoff macros require the input of two-level data set names, so we use a same column header name 'DataSet' in both metadata sheets.
- **Design a flexible structure and formats to cover unpredictable scenarios.** There are always many decisions in designing flexible metadata; for example:
    1. Do we expect all input data sets to come from a single or multiple directories?
    2. Do we process all input data sets independently or should they follow a certain dependency order?
    3. Do any input data sets need to be subset, and if so, is the subsetting dependent upon other data set(s) or fully defined by its own set of variables?
    4. When specifying data values, do we refer to a variable name, actual data value(s), and / or a code segment such as a programming function?

  A well-designed departmental macro (and metadata when using the MD approach) should always create the required outputs without needing post-macro fixes for studies that use it. However, in extremely complex study designs, this could be challenging because some study-specific requirements may simply not fit into an executable metadata structure or may not be anticipated during macro development. With this potential challenge, we want to design the metadata to allow optional pre-processing of input data prior to the macro call so that the macro functionality is still well-defined and applicable for all studies. This metadata design concept makes macros potentially useful for more studies, including complex ones which may be unpredictable at the time of macro development.
- **Metadata must be formatted as program-executable data values.** Traditional metadata is a non-operational living document, often containing rather free-style textual specifications. In the MD programming process, metadata eventually becomes part of program logic and thus needs to be in an executable format (i.e., each data value from a metadata table can be a valid code segment in the expected format to allow it to be incorporated into the macro programming techniques). For example, if a PROC SQL procedure is assembled automatically from metadata, the specifications that support this setup must be PROC SQL-executable code segments such as valid data set or variable names, functions, and where clauses.

MACRO PROGRAMMING CONSIDERATIONS

In an MD macro design, a key portion of the macro code is driven by study-specific metadata. The idea is that the macro code automatically reflects updates on metadata without changing its code. Below are a few important techniques and considerations we recommend for MD programming.

- **Read in Excel metadata directly using the SAS© Excel engine.** We use a departmental macro to read in any Excel spreadsheet (.xlsx file) directly using the SAS Excel engine so that users may reference and use the Excel sheets and columns as if they are SAS data sets and variables, respectively. It also checks for any non-ASCII characters in the data values and alerts users to

correct them. That macro is used by all MD macros to simplify the process: users can maintain the Excel metadata without having to convert them into SAS data sets prior to calling an MD macro.

- **Perform quality checks on metadata.** Metadata is read in by the macro and eventually becomes part of the macro code. Metadata quality directly impacts macro code and functionality. Besides implementing a process outside the macro itself to review and test metadata, a best practice is to have the macro itself perform metadata quality checks where feasible to alert users whenever it identifies any issues. For example, the macro can check for required missing or non-missing patterns on Excel columns and rows, or confirm a data set name has two levels as required.
- **Convert metadata into local macro variables for further programming reference.** Storing individual data values from the metadata Excel sheet in local macro variables allows them to be referenced and used separately in programming code. There are two options for this step depending on the programming design at the subsequent steps:
    1. Convert all columns and rows at the same time, e.g., using macro variables mv_dataset_1 to mv_dataset_<n> to store the data set names from the "DataSet" column and macro variables mv_subset_1 to mv_subset_<n> to store the where clauses from the "SubSet" column, where n is the total number of rows; or,
    2. Convert all columns across the current row within a %DO loop, e.g., using macro variables mv_dataset to store the data set name from the "DataSet" column and macro variable mv_subset to store the where clause from the "SubSet" column at the current row under processing.
- **Use more flexible programming techniques.** MD design is a more dynamic type of programming where some techniques may work better in providing greater flexibility than others. A typical example is to use PROC SQL instead of a data step to select data records and variables that may involve a data merge, if the process is driven by metadata.
- **Check outputs to confirm metadata quality if applicable**. Sometimes metadata quality issues can also eventually be reflected in the outputs that are generated by the macro. In that case, generating the required outputs is not the end of macro programming; a further step can be done to check for potential issues on outputs and alert users to correct such issues. For example, a certain missing data pattern in outputs may indicate potentially incomplete metadata components, which may not be easily discovered from a manual review of metadata. E.g., if there are 50 raw data sets in the source directory, but the MD macro outputs only 45 filtered data sets, that indicates the rest of five data sets may not be specified in the metadata.

APPLICATION - THE RAW DATA CUTOFF EXAMPLE

BACKGROUND AND GENERAL ALGORITHMS

An operational and analysis requirement for many interim analyses is to include only clinical trial data up to a pre-determined calendar date (i.e., the data cutoff date). This process helps to obtain a static and 'clean' set of data for the analysis. It is often referenced as the 'raw data cutoff' process as it is performed prior to SDTM mapping.

Although the general concept of only including data on or before the cutoff date applies to all studies, the more detailed set of cutoff criteria besides the cutoff date itself is usually study and data set-specific. Table 1 below describes general cutoff rules and potential differences between studies and groups them into several distinct dependency categories (i.e., 'Process Order').

As we can see, the general rules in the third column can be standardized in programming; at the same time, the anticipated differences between studies in the fourth column are complex. In addition, there might be rare cases where data sets do not follow the general rules (see an example in Table 2 row 5); in that case, pre-processing such data sets before passing them into the macro will be necessary.

Table 1. Raw Data Cutoff General Algorithms

Category	Process Order	General Rules	Differences Between Studies
No cutoff needed	0	Include all records from a given data set	• Data set names
Identify subjects to be included	1	Include subjects from a given data set where a milestone date (e.g., enrollment date) is on or before the cutoff date There is one and only one subject-level data set in this category	• Name of the subject-level data set • Variable name and data type of the milestone date
Identify records to be included; depends on the output data set from Process Order 1	2	Include records from a given data set for those subjects identified in Process Order 1 and: • No further cutoff, OR • The data occurrence date (e.g., event or intervention start date) has a missing value or is on or before the cutoff date	• Data set names • Variable name and data type of the occurrence date in each data set, if applicable • Whether or not this step creates a final output or is instead an interim step leading into Process Order 3
Identify records to be included; depends on the output data set from Process Order 2	3	Include records from a given data set only if they match records in other data set(s) coming in from Process Order 2	• Data set names • Merged data set names for a given data set • Where clause of data merge for a given data set • Whether or not this step creates a final output or is instead an interim step leading into Process Order 4
Identify records to be included; depends on the output data set from Process Order 3	4	Include records from a given data set only if they match records in other data set(s) coming in from Process Order 3 This category is rarely used as the majority of data can fit into the earlier categories	• Data set names • Merged data set names for a given data set • Where clause of data merge for a given data set

METADATA DESIGN

Table 2 below uses a few data examples to illustrate the structure and formats of the study-specific metadata that drives the programming of a raw data cutoff setup. In this study, the macro needs to process all data sets in the "raw" directory and output the final filtered data to &out_lib directory based on specified cutoff rules. The "raw" directory is specified in this study-specific metadata as part of the two-level input data set name in the "DataSet" column; the output directory is controlled by users through a macro parameter named OUT_LIB. The study-specific metadata is stored in the following Excel sheet columns:

- **Process Order** is the dependency order in the cutoff process, where order 0 can be processed at any time and orders 1, 2, 3, and higher are processed sequentially from low to high (see Table 1).
- **DataSet** is the input data set to be cut using two-level naming to allow different data sources; e.g., input data may be from the raw or the work library (in case pre-processing is needed).
- **DataSet2** contains one or more data set names to be merged with the input data set in "DataSet" for steps with order 3 or higher. It also uses two-level naming to allow different data sources between data sets; e.g., data could be from the final output library &out_lib or the work library.
- **Subset** is the where clause used in the PROC SQL data merge. The data set from "DataSet" is named "a" and data set(s) from "DataSet2" is named "b" ("c", "d", and so on if multiple ones) in PROC SQL.

- **ISODate** is the ISO date variable to be compared with the cutoff date. It may also instead be a function to convert a numeric date variable to ISO format.
- **DataSet_Out** is the filtered two-level output data set name to be created by the macro. A blank value in this column defaults the output data set name to be &out_lib..<DataSet>.
- **SUBJID** is the subject ID variable name in the input data set "DataSet", which could be different between data sets. This column is not displayed in Table 2.
- **StudyID** is the study ID for a given study. Although study ID does not change between data sets, we keep it in the metadata sheet as key information instead of passing it through a macro parameter. This column is not displayed in Table 2.

Table 2. Study-specific Metadata Structure for the Raw Data Cutoff

Row #	Process Order	DataSet	DataSet2	Subset	ISODate	DataSet_Out
1	0	raw.random				
2	1	raw.enrol			put(ENRLDATE, yymmdd10.)	
3	2	raw.aeyn				
4	2	raw.ae			STARTDTC	
5	2	work.ex_date			DOSEDTC	work.ex_date_incl
6	3	raw.ex	work.ex_dose_incl	a.SUBJECT=b.SUBJECT and a.VISIT=b.VISIT		
7	3	raw.imae	&out_lib..ae	a.PT=b.PT and a.EVENTID=b.AEID		

Table 2. Study-specific Metadata Structure for the Raw Data Cutoff (Continued)

Row #	What Does the Macro Do Given the Metadata?
1	The macro copies raw.random to &out_lib..random without further processing.
2	The macro creates &out_lib..enrol from raw.enrol where ENRLDATE is on or before the cutoff date. Note that a PUT function is used in metadata to convert numeric ENRLDATE to ISO date format.
3	The macro creates &out_lib..aeyn from raw.aeyn where subjects are included in &out_lib..enrol.
4	The macro creates &out_lib..ae from raw.ae where subjects are included in &out_lib..enrol and raw.ae.STARTDTC is either missing or on or before the cutoff date.
5	The macro cuts work.ex_date where subjects are included in &out_lib..enrol and work.ex_date.DOSEDTC is either missing or on or before the cutoff date. The macro outputs a temporary data set work.ex_date_incl to be used in the next step on row 6. Note that this is an interim step for raw.ex. Cutoff criteria for raw.ex require using a derived date variable DOSEDTC. Derivation of DOSEDTC is study specific and complex; it does not fit into this executable metadata structure. So, the user pre-processes raw.ex to create a temporary data set work.ex_date which contains the same data from raw.ex plus the new variable DOSEDTC. Then work.ex_date becomes the input data in this step for data filtering based on DOSEDTC.
6	The macro cuts raw.ex by merging it with work.ex_dose_incl where the visit is included in the latter data set. It outputs the final filtered data set to &out_lib..ex.
7	The macro cuts raw.imae by merging it with &out_lib..ae where the event ID is included in the latter data set.

MACRO PARAMETERS

With the design of metadata displayed in Table 2, we only need a few parameters to let users input additional basic information, which are listed below:

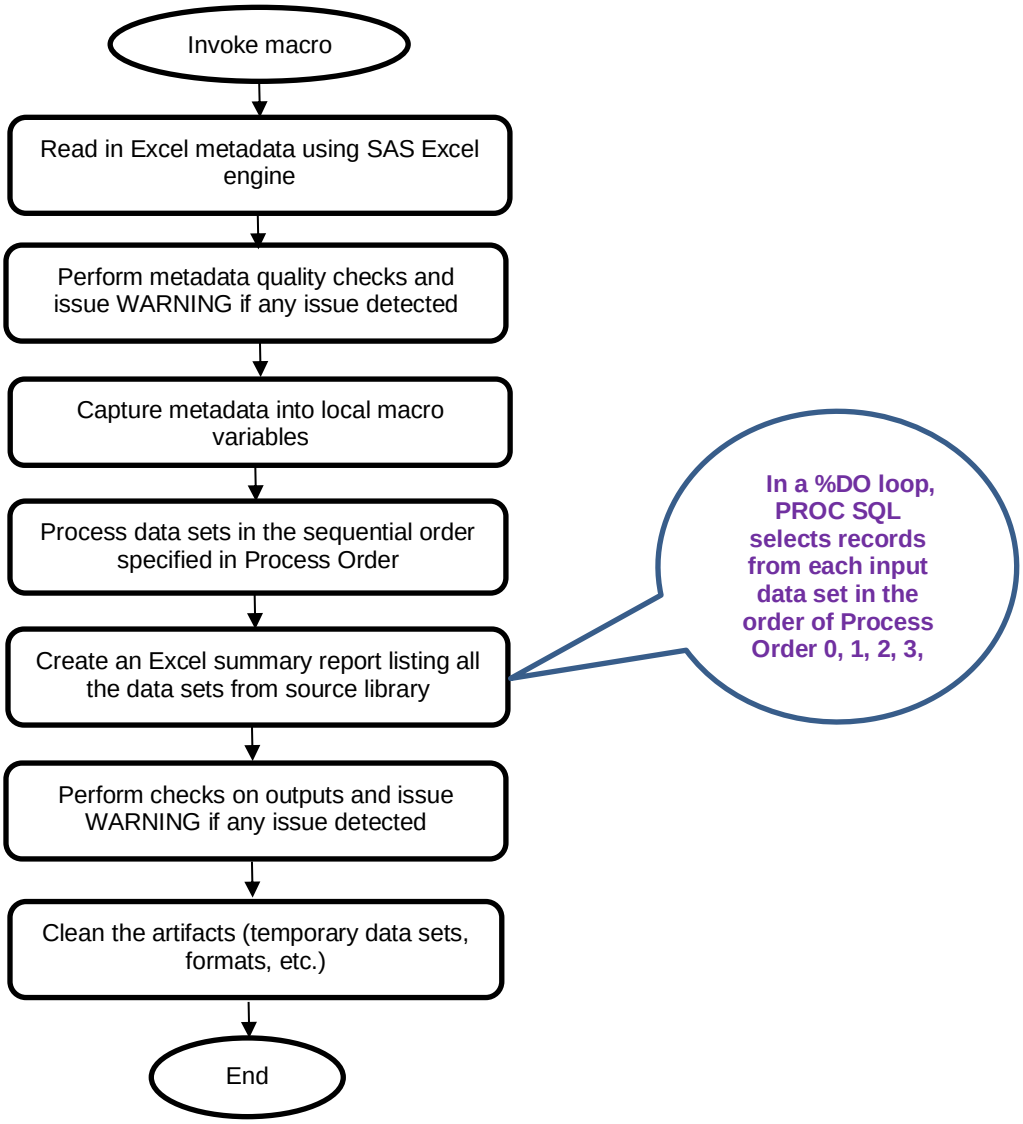
- **METADATA_DIR**: the directory path where the metadata Excel file resides. A default value can be defined based on a departmental standard directory structure.

- **METADATA_FIL**: the metadata Excel file name including the extension .xlsx. A default value can be defined based on standard file naming.
- **METADATA_SHT**: the metadata Excel sheet name. A default value can be defined based on standard sheet naming.
- **IN_LIB**: the library where the source raw data sets reside and in which a summary report will be produced that shows # records kept and dropped from each data set. A default value can be defined based on a departmental standard directory structure.
- **OUT_LIB**: the library in which to output the data sets with records included (i.e., not cut) for the analysis. A default value can be defined based on a departmental standard directory structure.
- **OUT_EXCL_LIB**: the library in which to output the data sets with records excluded (i.e., cut) for the analysis. A default value can be defined based on a departmental standard directory structure.
- **CUTOFF_DTC**: Data cutoff date in ISO format for a given interim analysis.

If a study is set up using a departmental standard directory structure and Excel file naming, in a typical macro call, users can take advantage of default parameter values and only need to input the data cutoff date. A simple macro call may look like **%m_rawdata_cut (cutoff_dtc=2020-03-19)**!

MACRO PROGRAMMING FLOW

The following diagram shows the high-level programming steps within the macro:



CONCLUSION

Metadata-driven programming, with a careful design of metadata and use of efficient programming techniques is a much more dynamic approach for departmental macros, especially when targeting areas where differences between studies are complex and less predictable. This approach further standardizes programming for more complex data processing such as the raw data cutoff and other pieces in SDTM and ADaM. It greatly simplifies user input using fewer and simpler parameters. It cuts down tremendously on maintenance efforts and macro re-testing when new input values are encountered, thus facilitating continued and instant user access to the power of the MD macros without having to take them back into the shop for parameter maintenance and upgrading. It throws the door wide open for a much more robust and stable macro library at department level!

ACKNOWLEDGMENTS

Thanks to the Biometrics macro working group in Seattle Genetics, Inc., especially Jinit Mistry and Yinghui Wang, for their valuable input on the design of a series of metadata-driven macros.

CONTACT INFORMATION

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

Ellen Lin
Seattle Genetics, Inc.
1-425-760-2248
ellin@seagen.com