PharmaSUG 2012 - Paper PO13

Creating Metadata Driven Macros to Gain Dynamism and Reduce Complexity in Macro Calls.

Fernando Enriquez, PharmaNet/i3, Mexico City, Mexico

ABSTRACT

It often happens in clinical programming that we have to create standard derived data sets for several studies. It is always a good idea to create a macro and use it for all studies. The problem comes when there are too many parameters to consider for the study; large lists of changing values for parameters to be defined in the macro call, variable names changing in source data sets across studies, different categories to be derived or different value ranges for creating categories in the derived data sets. Many more problems can be listed and here is where metadata is useful to resolve such complexity when creating macro calls.

Our metadata is a plain text file structured as a table in which the main macro can import as a data set to obtain all the parameters needed to process the derived data set with personalized specifications depending on the study; such values can represent categories to be processed with value ranges, lists of values, variable names, derivation specifications, filters, etc. This will give the main macro more dynamism and the macro user will find it easier to adjust the metadata file as needed without touching a single line of code - not even the macro call.

This paper is directed towards SAS® Base programmers with some SAS Macro Language knowledge.

INTRODUCTION

The dynamism that a macro can give us to avoid recoding, save time, keep processes standardized and many other advantages can be affected when the source data sets are not standardized. Other considerations should be made when programming a macro to be used in several studies because, requirements will change across studies. This is when the use of metadata can make our programming more efficient by getting parameter values or deriving specifications through importing the metadata file into a data set.

One important thing to note is that we can have as many metadata files as needed. For example, we can have a metadata file with a list of values, flags, source data set names and variable names; alternatively, we can create a metadata file with value ranges, category definitions and derivation specifications. Such lists of complex parameter values can be easily set up in a metadata file instead of defining them in the macro call. This can be more difficult to do if the user does not know the macro parameter specifications, or if the user does not understand the macro code.

CASE STUDY

During the next sections a simple case study will be used to illustrate the metadata-driven macros approach. A sample reactions derived data set will be developed during the explanatory topics in this paper. Please note, this example is very simple and has been chosen just for illustrative purposes. This approach is, however, recommended for complex derivations where there are too many considerations requiring maximum flexibility in the macro programming and many parameter declarations.

FIRST GLANCE

This project needs a derived data set containing information on treatment reactions provided by patients through an electronic diary. This diary is filled out by patients after receiving each study drug dose as scheduled in the protocols. Patients should provide information from day 1 to day 7 following drug administration. The clinical study requires safety analysis of these reactions for all of its protocols. Here are some of the considerations:

- Diary data is collected by different vendors, so data comes in different formats and structures for each of the protocols.
- Protocols target different populations; hence some of the ranges and derivations for analysis change.
- Different lists of reaction information are collected for each of the protocols.
- Descriptive reaction information from the vendors is different from the project descriptive reaction information. Synonyms, letter case, use of plural or singular, etc. are some of the details which complicate the incorporation of vendor data into the project database.

The first step is to identify the needs of the derived data set. In this case study, we want to summarize all of the reported and not-reported diary information, including reaction description, visit numbers, day numbers, day dates, start date, stop dates and duration of the reactions. With this objective in mind, it is time to identify the dynamic parts to be included in the macro in order to create standard output regardless of the data source and study specifications.

This is where pseudocode is used.

USING PSEUDOCODE AS THE STARTING POINT

Use pseudocode for sketching out the structure of the macro program before the actual coding takes place. This will help to identify the parts that will be dynamic in the programs (processes, variables, ranges, etc.), and will be included in the structure of the metadata file. The following illustrates some examples of pseudocode:

Example 1:

```
/*Transpose vendor's data if it comes in a vertical structure*/
If macro variable "structure" = "vertical" then transpose data in order to have one record per patient/visit/day/reaction and sort it using the same variables in order to standardize the rest of the processing.
```

Example 2:

```
/*Read data sets*/
Read vendor's data set keeping key variables and list of variables stored in macro variable "vars" gotten from metadata in order to derive just what is needed for specific study.
```

Example 3:

```
/*Incorporate vendor diary information in study database*/

Sort vendor data set and merge it with metadata to incorporate a link between vendor data set and study data set.

Sort study data set and merge it with metadata to incorporate a link between vendor data set and study data set.

Merge vendor data set with study data set using the already created metadata link.
```

Example 3 is a clear case where metadata can work efficiently by providing a link between vendor data and study data sets. Once this is incorporated in the design of the metadata, it is very clear and easy to establish this link, while conventional macro calls using ordinary parameters can be confusing. An example will be described in the next section.

Once you identify all the potential dynamic parts in the processing, then the metadata can be built together with the actual macro code. Keep in mind that macro coding and creation of the metadata files will take more time to complete than conventional programming, but will save time when they are used to produce output for several studies.

BUILDING THE METADATA FILE

Now that we have a full picture of the macro and its needs, it is time to build the metadata file. The metadata is a plain text file, delimited by either tabs or special characters. Its structure is similar to a data set. The first row is used to define the variable names and the following rows contain the values. The file should be saved with a descriptive name and extension, such as variables.meta. Using the reactions data set as an example, the metadata file structure can be something like this:

Example 4:

```
var    |study_desc    |vendor_desc   |label        |type   |flag
chls   |chills        |Chills        |Chills       |se     |1
swe    |swelling      |Induration    |Swelling     |se     |0   → swelling vs Induration now linked
red    |redness       |Erythema      |Redness      |lr     |1   → redness vs Erythema now linked
```

Example 4 illustrates how processing of inconsistent data from the vendor and study data are simplified with the metadata structure because now equivalent values are described in a flexible file, which makes macro processing very simple and intuitive. Implementing this in conventional macro calls would be confusing and the macro calls could look overwhelming.

If variable names change for other studies, you can just adjust them as follows. Please note if you don't need to derive something already declared in the template, a Boolean flag is enough to indicate that, as illustrated in Example 5.

Example 5:

var	**|study_desc**	**|vendor_desc**	**|label**	**|type**	**|flag**	
chils	|chills	|Chills	|Chills	|se	|1	**→ flag set to 1 to derive this reaction**
indur	|swelling	|**Induration**	|Swelling	|se	|0	**→ flag set to 0 when no needed**
ery	|redness	|**Erythema**	|Redness	|lr	|1	**→ flag set to 1 to derive this reaction**

Once we have the metadata file structure, it can be used as a template to replicate for each study. Note that while you can create metadata using other file format, such as a spreadsheet, for this paper's purpose, a delimited plain text file is used to make things easy to understand.

Remember to document the metadata files. You can create a separate metadata file to describe the whole set of metadata files that created for your project. That way, any independent programmer can easily understand how to configure the metadata files for a certain study.

READING METADATA FROM THE MACRO

Now that we have the macro pseudocode and metadata structure in place, actual coding can occur. One of the most important things when coding the macro is the incorporation of metadata files into the SAS environment. One example of reading the metadata is the described in Example 6.

Example 6:

```
PROC IMPORT DATAFILE="variables.meta"  DBMS=dlm OUT=variable_meta REPLACE;
  DELIMITER="|";
  GETNAMES=yes;
RUN;
```

Once the metadata values are imported to the WORK library (or any other defined library) as a SAS data set, it can be used to get all parameter values required for the macro to process the derived data set. You can use this data set to create merges, get variable names, lists of items, etc.

Let's say the ranges metadata was created and imported We can easily use information from the SAS data set to obtain the required values and define several macro variables to be used in a process within the macro, as shown in Example 7.

Example 7:

```
PROC SQL NOPRINT;
   SELECT MIN(lower), MAX(higher) INTO: lower, :higher
   FROM ranges_meta;
QUIT;
```

CREATING THE MACRO CALLING PROGRAM

Once the macro program and metadata file(s) are ready, the last element required is the macro calling program. This consists of a simple macro call line without parameter values, as shown in Example 8.

Example 8:

```
%create_reactions_dataset();
```

BASIC METADATA-DRIVEN MACRO STRUCTURE

Once all elements are in place (the actual macro program, the metadata file(s) and the macro calling program) a basic structure can be implemented so that the maximum benefits are obtained.

- Save the macro program in a general folder which can be used for all studies.
- Create a program in a single study, containing the macro call (no parameters defined in main macro).
- Save the metadata templates in a general folder and copy it to a single study and define values as necessary, this structure makes it easy and intuitive for macro users to define parameter values.

CONSIDERATIONS

- Use this approach only when you anticipate re-use of the macro for multiple clinical studies.
- Use this approach only when you anticipate complex dynamism in the macro which would require a lot of parameter definitions.
- The macro program and metadata work closely together, hence a good analysis when creating the pseudocode will lead to flexible and complete metadata an efficient re-use of macro code.
- Follow good programming practices when coding the macro. For instance, detailed comments in the program are very useful when an unfamiliar programmer needs to update the code.
- And independent and original validation program should be written for every output release to catch errors when configuring the metadata or bugs in the macro code.
- Even though more that one metadata file can be created for the macro processing, try to create no more than two or three metadata files to cover your needs. Otherwise, the complexity in conventional macro calls will be transferred to this metadata structure. Keep things simple.

CONCLUSION

Having a simple metadata-driven macro structure in place will save time for your team, and will make the team's work easier. Macro experts are no longer needed to create an standardized data set, and even people who do not know the programming language can easily set up the metadata file and produce the data set without modifying a single line of code!!!.

ACKNOWLEDGMENTS

Thanks to PharmaNet/i3 who gave me support for presenting this poster and facilitated everything needed to present and attend PharmaSUG 2012 in San Francisco, CA.

Thanks to Nancy Brucken for giving me answers to my questions about dynamics and details for submitting this paper, and for her suggestions in final paper submission.

Thanks to Brad Danner for sharing his experiences in poster design from previous PharmaSUG events.

Thanks to the Mexican Team and Mark Matthews at PharmaNet/i3 who took the time to check this paper and related poster to provide their opinions and advices.

RECOMMENDED PAPERS/PRESENTATIONS AT PHARMASUG 2012

If you read and are interested in this paper and related poster, you also might be interested in:

- Data Standards and Quality: Building Flexible ADaM Analysis Variable Metadata by Songhui Zhu & Lin Ya.
- Industry Basics: Metadata: Some Fundamental Truths by Frank DiIorio.
- Techniques and Tutorials: Building the Better Macro: Best Practices for the Design of Reliable, Effective Tools by Frank DiIorio.

CONTACT INFORMATION

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

Name: Fernando Enriquez
Enterprise: PharmaNet/i3
Address: Platon Edificio 64 Departamento 002
City, State ZIP: Ecatepec, Estado de Mexico 55018
Work Phone: +52 (55) 2593 9398
E-mail: fenriquez@pharmanet-i3.com, fernando_hispano@hotmail.com
Web: http://mx.linkedin.com/pub/fernando-alonso-enriquez-bocardo/45/90/4a1
Twitter: @ferhispano