Human Efficiency: A Factor That Can’t Be Ignored When Designing a SAS® Macro

Michele Zhang, Merck & Co., Inc., Blue Bell, PA

Abstract

In the pharmaceutical industry, the most common and complex clinical trials are the phase II and phase III trials. Projects involving multiple phase II and phase III trials are particularly complex. For programmers working on these projects, one of the major responsibilities is to produce safety and efficacy tables to support data analyses. In order to speed up this programming effort, we often use macros to program the same type of tables across different protocols. However, the way we design our macros can make a huge difference in our daily programming work. This paper uses a case study to demonstrate how we approach our real-life problems and how we have altered the traditional method for designing macros in order to make our programming work more efficient.

Introduction

As we prepare NDA submissions, tables are usually programmed, tested and validated before a database is frozen. However, because of the internal review process, changes are often necessary even after a database is frozen. Most of these changes are cosmetic, dealing primarily with titles, headers and footers. One way to handle such changes is to go into each validated program or macro and make the changes accordingly. Although these changes might be simple ones, the number of programs and macros involved could turn out to be quite significant. If there are many changes, the process becomes overwhelming and difficult to control. There is also an increased risk of errors and inconsistencies among team members, reducing the comfort level in the validation or QA (Quality Assurance) process. In some critical programs, it is even necessary to complete extra validation. This process is very tedious, time consuming and inefficient.

Is it possible to provide a more efficient way to perform these changes? Can we design a more flexible macro to handle these issues? Can we make these changes without touching an already validated macro? More specifically, in a final CSR (Clinical Study Report), there may be many titles, headers and footers that are the same or that have the same portion of text. Is there any way to centralize this information to quickly, efficiently and flexibly support any request for changes from a statistician under time pressure? This paper uses an example involving subject characteristics tables to compare the traditional programming method with the alternative approach.

Traditional Method

Program flow
When we use a macro to handle the same type of tables across multiple protocols, our traditional programming method for the macro normally involves the following four major steps:

- Read and prepare data
- Calculate and transpose data
- Generate data sets or reports
- Convert data sets into rich text tables

In this paper, our focus will be on the last step. One of the utility macros that we use to create tables is the %rtftable macro. This powerful macro was developed internally at Merck to convert SAS data sets or reports into highly customized tables in rich text (RTF) or Microsoft Word (DOC) format. The function and design of this macro were presented at the 2003 PharmaSUG Conference.

To convert data sets into rich text tables, we normally define titles, headers and footers before calling the %rtftable macro. Below is the typical code that is inside our macro during this step.

Example code

```sas
/* Define protocol xxx1 start */
%if &prot=xxx01 %then %do;
    /* Define Titles */
    %LET TITLE1= Summary of Subject Characteristics;
    %LET TITLE2 = by treatment Group;

    /* Define Headers */
    %LET HEADER1  = | &treatment        |    Placebo |  Total |;
    %LET HEADER2 = |      |     n  (%)         |  n  (%)         |  n  (%)|;

    /* Define Footnotes */
    %LET FOOTER1=Percent is computed as 100*(n/N).;
    %LET FOOTER2 = number of subjects who have the indicated characteristic.;

/* Define protocol xxx1 end */
%end;

/* Define protocol xxx 2 start */
%if &prot=xxx02 %then %do;
    ...........
    ...........
/* Define protocol xxx2 end */
%end;

/* Define protocol xxx3 start */
%if &prot=xxx03 %then %do;
    ...........
/* Define protocol xxx3 end */
%end;
```

2
Problems with multiple protocols

Although we use the %rtftable macro to produce the same type of tables for multiple protocols, we often need to define different titles, headers and footers for these protocols. As indicated by the example code above, this information is normally defined before calling the %rtftable macro. Since all of this code is then inside the macro, this traditional macro design creates the following problems.

- A less dynamic macro: Any changes to titles, headers and footers require changing the macro, which increases the contents of the macro and reduces its benefits.
- Repeated validation: Changes reduce confidence in the validated macro, resulting in an endless validation effort.
- Extra coordination: Since different programmers and statisticians work on different protocols, additional effort is needed to maintain consistency on final reports.
- A time-consuming process: Dealing with changes to the macro adds time pressure in meeting the final submission deadline.
- Duplicated versions: In large projects, protocols roll out one by one, and multiple macro versions are needed in order to make changes.

Some of these concerns become even clearer through a schematic representation, as in Figure 1 below. This diagram shows how the traditional programming method works in handling the creation of the same type of tables for multiple protocols, as well as for defining the titles, headers and footers for these protocols.

Figure1
Macro design perspective

Figure 1 demonstrates how the macro function reduces the need for duplicated code when programming the same type of tables for different protocols. This is the major benefit of designing a centralized macro. However, the existence of protocol-level code makes this centralized macro less dynamic. For example, as represented by the diagram and by the example code defined previously, the texts of different protocol titles, headers and footers are defined inside the macro before calling the \%rfttable macro. Since these texts are changed frequently by different protocol statisticians or reviewers during different stages of a study, storing this portion of code inside the macro program causes the types of problems described above.

Communication perspective

Figure 1 also clearly indicates that, the traditional programming method is not a good Model for communication among different groups of statisticians and programmers because there is no direct link between each of these groups. In fact, the model itself can actually increase the traffic between the protocols and the macro, while providing no opportunity for coordination or communication among the team members. Such problems increase the risk of error and inconsistency among the final reports. In particular, given the nature of the changes in large projects with multiple protocols being completed one after another, there is no good way to make changes without touching the macros. Repeated changes then result in duplicated macro versions. Multiple protocols, multiple macro versions and multiple team members without adequate channels of communication easily combine to create inconsistency among the final analysis reports, which play a vital role in getting new drug approval from the FDA.
Alternative Method

Can we still get the benefit of a centralized macro and also provide the flexibility to make changes on a protocol level? Can we change the model or process to enhance the communication between different team members?

Concept of the new method

The concept of the alternative method is to remove less dynamic code from the centralized macro and to reduce any unnecessary work by storing frequently changed text in a new centralized location. Figure 2 below displays how this alternative process operates.

Figure 2

Macro design perspective

In this case, the titles, headers and footers are frequently changed text that tends to be more specific to individual protocols. Storing this hard text code in the centralized file location reduces the time spent on unnecessary, repeated macro code work or validation brought on by changes in the macro. Therefore, when these unpreventable changes are moved out of the macro and into this separate file, the result is a more dynamic macro.

Communication perspective

While one aspect of the alternative method is to remove text code from the centralized macro, the other aspect is to store this code in its own centralized location. This centralization can reduce repeated coding on different macros in the same protocol or
repeated coding in the same macro across different protocols. Figure 2 also displays the
link from the central file to different protocol team members. This model provides
excellent channels for coordination and communication among different groups, which
reduces the risk for any potential inconsistencies or errors.

The centralized file includes two parts for handling information. One part involves
centralizing common information among different protocols or common information
among different macros. For example, the treatment or drug name can be shared by
different protocols if the protocols are using the same compound, whereas the MITT
population definition is shared by different macros within the same protocol.
Centralization reduces repeated work on both the macro or protocol level. Below is an
equivalent example of a centralized file.

Example of a centralized file

/* define Globe text or common Titles/Headers/Footers */
/* Titles shared by tables */
%let trt=by Vaccination Group;
/* Headers shared by tables */
%let head=Treatment ABC;
/* Footers shared by tables */
%let bigN_ = N = Number of subjects randomized to the respective
vaccination group who received at least 1 injection.;
%let MITT1 = This population includes all subjects who received all 3
vaccinations and are seronegative at Day 1 and PCR negative on Day 1
through Month 7 for Treatment ABC. Endpoints are counted starting after
Month 7.;

/* define protocol level Titles/Headers/Footers */
/* Define protocol xxx1 start */
%if &prot=xxx01 %then %do;
/* Define Titles */
%LET TITLE1= Summary of Subject Characteristics;
%LET TITLE2 = by treatment Group;
/* Define Headers */
%LET HEADER1 = | &treatment | Placebo | Total |;
%LET HEADER2 = | n (%) | n (%) | n (%) |;
/* Define Footnotes */
%LET FOOTER1=Percent is computed as 100*(n/N).;
%LET FOOTER2 = number of subjects who have the indicated characteristic.;
/* Define protocol xxx1 end */
%end;
/* Define protocol xxx2 start */
%if &prot=xxx02 %then %do;
...........

The second part only involves storing information across different protocols. In this situation, users can choose to store each table’s titles, headers and footers in the central file location or in their protocol level calling syntax programs. Below is an example of macro call which has all titles, headers and footers defined inside the calling program.

Example of macro call syntax

```
%LET TITLE1= Summary of Subject Characteristics;
%LET TITLE2 = by treatment Group;

%LET HEADER1 = |      | &treatment    |    Placebo   |  Total  |;
%LET HEADER2 = |      |    n  (%)     |  n  (%)      |  n  (%) |;

%LET FOOTER1=Percent is computed as 100*(n/N).;
%LET FOOTER2 = number of subjects who have the indicated characteristic.;

%sub_char ( prot      =,
            n_title   =2,
            n_footer  =2,
            n_header  =2,
            n_lfoot   =1
          );
```

Advantages and disadvantages of the alternative method

Advantages

- Efficiency: it reduces duplicated coding and repeated work for any changes in the macros involving titles, headers and footers
- Consistency: it minimizes the risk of any inconsistency in the output among different team members
- Flexibility: it provides a flexible way to make changes without changing the main validated macros
- Time savings: it requires less work and fewer changes in code
- Accuracy: it improves the confidence level of centralized macro validation
- Convenience: it provides a complete list of table titles that might be very useful for preparing a Stat Review Aid package

Disadvantages

- Coordination: extra effort is needed to coordinate team members working on the centralized file
Conclusion

Overall, the alternative method provides many solutions for the problems that we have faced with our programming work. It not only provides an efficient and flexible approach designing and handling changes for the macro, but it also improves our work process, programming efficiency and quality control. Most importantly, it incorporates the basic programming principle: separating the hard code text from the program logic algorithm.

Acknowledgments

The author thanks Allan Glaser for his continuous support and valuable comments, Liping Zhang for utilizing the concept in their project and useful feedback, and Asam Ellen for reviewing the paper and her variable comments.

References


Contact Information

Your comments and suggestions are valued and encouraged. Contact the author by mail or e-mail at the following addresses:

Michele Zhang
Merck & Co., Inc.
UN-A102
Blue Bell, PA 19422

Work phone: 484-344-7869
Fax: 484-344-7651
E-mail: zhang_michele@merck.com

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.