PharmaSUG 2019 – Paper BP-194

Macro Templates - Industry Specific SAS® Programming Standardization

Tabassum Ambia, Alnylam Pharmaceuticals, Cambridge, MA

ABSTRACT

Industries are increasingly focusing on developing their own sets of SAS® macros to support development of datasets (SDTM, ADaM), Tables, Listings, Figures (TLFs) and proceeding towards their own standardization. The use of job-specific standard set of macros affects production time, resources required, programmers' learning opportunities, debugging time and the quality of output. There are pros and cons of using standardized macro templates. This is a quasi-automated process which largely cuts down the effort and amount of time required for programming that helps to complete a huge amount of work within a short time frame, outputs remain consistent and reduce probable programming errors. Stringent use of standardized automated macros also has its challenges which includes limitation of outputs different from standard ones where programmers need to circumvent macros, time required to debug errors, training required for new programmers etc. Being entirely dependent on these macros for a long duration may limit the logical thinking and open coding skills of programmers in developing outputs independently. These skills are essential to remain open to a wide range of programming needs. This paper discusses different aspects of the extensive use of industry standardized macros.

INTRODUCTION

The fast-paced industry of drug research requires a large amount of work to be completed within a constricted timeline to facilitate the ongoing innovation. This process involves the generation of datasets, tables, listings and figures to display the results of analysis. In general, these outputs are similar in nature with the exception of a few study-specific unique ones. In order to reduce time and efforts, and avoid repetitive work, industries are moving towards standardization of programming technique by creating their own sets of macros and program templates to generate standardized outputs.

HOW A MACRO WORKS

As we all know, a macro is a programmable pattern which allows the implementation of multiple steps of instruction, that expand automatically, and execute a set of codes to perform specific tasks without requiring repetitive commands. As defined in the macro, the set of codes executes sequentially and/or dynamically to produce a predetermined set of output. This pattern of coding allows a programmer to re-use the frequently used lines of codes in multiple programs. The macro language serves as a dynamic editor for SAS® programs. When SAS® compiles a program, it first sends the program to a word scanner which intercepts the macro syntax prior to reaching the compiler. The macro processor translates the macro syntax into standard SAS® syntax which is then compiled.

A macro starts with %macro code and ends with %mend statement. Generally, a macro is composed of macro variables and macro programs. The macro variables are defined within the %macro statement which keeps the values of these variables dynamic allowing them to be modified by a symbolic substitution. The simplest macro variables can be created with %let statements. Key words related to macro variables in the statement or program are preceded by a percent "%" sign while an ampersand "&" sign is used to reference simple macro variables. Once the execution starts, the statement which contains multiple commands with percent "%" sign determines the instruction or function which are to be executed within the step, and the variable names preceded by ampersand "&" sign capture the values defined in the final macro statement to use those certain values in the step.

STANDARD SAS® MACRO TEMPLATE

A standard macro template is a combination of several steps of codes that include multiple macros or open codes to complete a certain type of analysis or to generate datasets, tables, listings or figures. Usually the macro system or the template containing all the program steps generates a specified type of output display by taking the dynamic values of certain variables into account.

INDUSTRIES MOVING TOWARDS STANDARDIZATION

Pharmaceutical industries and Contract Research Organizations (CROs) process a huge amount of data in support of multiple projects to complete as expeditiously as possible. A single drug needs to undergo numerous clinical trials through Phases 1, 2, 3 and 4 in order to be made available to the population worldwide. Different clinical trials yield varying results involving a wide array of primary and secondary endpoints, based on which, the drug is approved for marketing. This entire process is rigorously stretched over an extensive duration - years and sometimes even a decade, prior to the submission for regulatory approval. The number and the level of complexity of projects that test drugs' safety and efficacy, through the analysis of demographics, common adverse events, laboratory results, vital signs etc. are growing. Consequently, the development of a standardized system of SAS® program templates with macros has grown to be extremely popular and more companies are inclining towards replacing the extensive manual programming with this trend of automation.

ADVANTAGES OF STANDARD MACRO TEMPLATES

There are a wide range of benefits of using standardized macro templates, which is the reason more industries are focusing towards standardization.

- ### FASTER

  Standardized macros are quasi-automated process which significantly cuts down the time and efforts enabling programmers complete a huge amount of work within a short time frame. They also help to develop predetermined set of outputs in a preferred format.

- ### LESS RESOURCES REQUIRED

  Since the time required for programming is significantly decreased, a small team can deliver a high volume of analytical output in formats specific to project type. This along with the previous factor cuts down the overall cost and makes management of tasks easier.

- ### PRECISION

  Standardized macros maintain consistency in layouts, presentation of outputs while reducing the chance of errors. Codes which determine the preferred statistics and display options are already embedded within the templates. Outputs are generally free from the risk of errors in calculations or maladjusted alignments unless any significant change is required in the standardized specifications of datasets and TLFs requiring macro customization.

- ### PREVENTS MONOTONOUS PROGRAMMING

  Standardized macros require less instruction and system resources to generate similar output across studies. Less repetition of similar code prevents monotony. Programmers do not need to run the same programs multiple times to generate datasets or TLFs of identical specifications with just a change in subset condition, or to type and re-type titles and footnotes. The entire series of codes may be executed without repetition with just few macro definitions.

CHALLENGES OF STANDARD MACRO TEMPLATES

This growing reliance on the utilization of standardized SAS® macros also has its limitations besides the advantages. The constraints of strictly using standardized macro templates are discussed in the light of a few real-life scenarios.

- ### LIMITS RANGES OF OUTPUTS

  When a study needs a customized layout of TLFs or uses analysis models that are not within the range of standardized macros, creation of outputs different from the standard may be a challenge. For example:

  1. **Restricted display:** If certain columns of a table are specified within the macro, additions of any extra columns based on the requirements of a new study may require extensive manipulation of the sections in the macro controlling display options. Display is generally performed by “proc report” in an open code. Programmers can define variables and control options easily in a “proc report” step. Often a standardized macro has only certain number of variables specified to be displayed. Programmers are only expected to fill out the dynamic parameter sections within the macro call without being able to see what is happening behind the macro. In those cases, addition of extra columns to the output would be challenging to programmers and is expected to require additional time affecting efficiency.

  2. **Sort limitations:** If extra sort variables are defined in the specification of TLFs and the standardized macro did not take those extra variables into account, the required sort may be difficult to manage. Sometimes sorts within standard macros are specified to consider only numeric variable and assign the corresponding character values in the outputs. Occasionally, the outputs require sorting by the character values instead of the numeric ones or the coded sort variables. In such cases, the macro may not yield the desired results. A common instance is sorting by character test names of laboratory tests (PARAM) instead of numeric values of laboratory test (PARAMN) or coded test values (PARAMCD). When sorted by PARAMN or PARAMCD, often the macro will have a standardized format section to capture the associated character values from the PARAM variable. If the output needs to be sorted by the values of PARAM instead of PARAMN, this may not take PARAM values into account through any straight forward manipulation technique. The reverse also raises similar concerns. If macro only considers character PARAM as the sort variable, sorting by PARAMN may generate errors and require further manipulation.

  3. **Miscalculation:** Sometimes standardized macros fail to generate appropriate counts. In my past experience, for an adverse event table intended to display counts of subjects per Body System or Organ Class (AEBODSYS), Dictionary-Derived/Preferred Term of Adverse Event (AEDECOD), Severity of Adverse Event (AESEV) and Causality/Relation to Study Drug (AEREL), the standard macro template generated precise counts per System Organ Class, Preferred Term and Severity. However, it failed to count unique subjects per Relation to Study Drug. The macro considered each occurrence of event as a unique record instead of counting unique subjects. Also, as both production and validation programs were using standardized macros, this did not appear as an error in the validation process. Manual coding was required later to generate appropriate counts.

- ### INCREASED ERRORS

  Errors may appear during manipulation of specified macros to generate outputs different from the standardized ones. From the second scenario specified in the previous section, if the standardized format macro is set to capture the character test values from PARAM according to the corresponding values of numeric PARAMN and only accounts for the numeric PARAMN variable, it should throw errors if PARAM is used instead of PARAMN. A simple solution may be the creation of another numeric variable per the desired sort of character PARAM variable and using that new sort variable instead of the direct use of PARAMN from analysis dataset. However, that runs back to manual programming again instead of having a standard macro work to create. Similarly, if the sort and format standard macros are set to use only character PARAM variable, but study requires the use of numeric PARAMN values, the macro will generate errors.

- **LONGER TIME REQUIRED**

  An increase in the time spent on programming is often observed which may be attributed to the number and duration of attempts required to debug errors while trying to manipulate a standardized macro template. In all the scenarios discussed above, programmers had spent quite a few hours, perhaps beyond what is desirable, to make updates through macros as they were strictly required to use the standardized templates.
  For the first scenario above, programmers were able to display few columns but without the proper alignment as controlled by the macro. Later they had to revert to using traditional "proc report" directly. For few other tables, the time required to explore through the macro to populate all desired columns could not be accommodated and the display was programmed manually at the last moment in rush.
  For the second example, they were able to create new sort variables as described in the preceding section. But since a new sort variable corresponding to the tests was created manually, the program needed updates following every data transfer due to new tests coming in. As this needed to be programmed using open code, the macro was not considered as effective as it was expected to be.
  In the third case, programmers had to program and derive the counts manually because the situation was deemed as a limitation of the standard macro by the technical team later on. Programmers had tried to make the macro work using different approaches spending a number of extra hours but ended up with no effective results. Later they had to switch to manual programming to derive the unique counts. If they had the flexibility right at the start to use both standardized macros/table program template and manual coding depending on what makes the process faster and precise, the process would have been more efficient. Given the fast-paced nature of drug research and submission process, the production of non-erroneous output with efficiency is paramount.

- **LACK OF READABILITY, REUSABILITY, PRODUCTIVITY**

  Standardized macros are useful, but usually more in the hands of the author of the macros. When passed on to other programmers, understanding and working through a different programmer's codes and approach may seem cumbersome and involve longer time to solve any issue or to use the macros effectively. Clear, extensive, documented training is required for programmers at all levels to be able to efficiently utilize standard macro libraries.

- **RESTRICTS IMPROVEMENT OF PROGRAMMING SKILLS**

  Sometimes if the programmers utilizing the macros do not have access to the original macro area to visualize or learn which codes are running in the background, they often need longer than the technical team members (who created the macros) to figure out the options to make the updates work. This hinders their learning process. If all programmers can see what is running behind the macros, they can be far more effective making the required updates. This would help improve their programming skills as well as contribute making further changes in the improvement of standardized macros for better accuracy and efficiency. Without being able to see the actual codes running, they cannot be as effective applying updates or help making the macro library more robust. Also, programmers may lose their coding skills if they are confined within such a strict macro environment for an extremely long duration. New programmers do not get to learn a

wide range of codes and programming techniques due to their lack of exposure to manual programming needs.
For instance, a new programmer who only ran direct macros to load titles and footnotes and other macros controlling display and formats without having much option to explore title, footnote and report options, may not be able to solve common challenges. For example, displaying units with "^" while "^" sign may have been used in "proc report" to split lines, changing the length or other properties of underlines in spanning headers, alignments or removing extra space in some titles and footnotes.

• RESTRAINS LOGICAL THOUGHT PROCESS

When a programmer tries to write open codes and think of several approaches to derive the desired components of an output (counts, frequencies, sort, merge etc.), they try to explore different programming techniques, search for better approaches and think further towards an effective process. These eventually lead to the improvement of logical programming thought process and better efficiency over time. Remaining entirely dependent on standardized macros without much exposure to a wide range of outputs different from the standard ones may restrict the logical thought process and open coding approach.

• LIMITS UNDERSTANDING OF CONCEPTS

When a programmer tries to program and derive outputs manually, they also tend to think and clarify their concepts further about datasets and TLF shells. If they run only standardized templates changing the dynamic variables, the practice does not provide enough exposure towards understanding new concepts. Programmers may actively choose to spend time to improve their conceptual learning but that is often deemed impossible given the fast-paced nature of the industry. Programmers need the opportunity and time to check with statisticians about what the TLF shells actually mean and what is desired to be displayed versus inexplicable use of a standard macro code library. Given the constricted timelines, programmers should not be tempted to use macro templates without trying to clarify basic concepts and running the programs to assure quality and meeting specifications.

As a reference, the last row and column of the shift table below needed clarification:

Table 1: An example of a laboratory shift table

Parameter Visit	**Baseline**				
Low	**Low**	**Normal**	**High**	**Total**	**Missing**
Normal	xxx	xxx	xxx	xxx	
High	xxx	xxx	xxx	xxx	
Total	xxx	xxx	xxx	xxx	
Missing					xxx

While programming this table, the production programmer was confused about the last cell at the bottom right corner whether to display unique counts of subject who:

- Had NORMAL baseline record but any ABNORMAL (Low or High) post-baseline, or
- Had baseline record but missed post-baseline records at all visits, or
- Present in Subject Level Analysis Dataset (ADSL) but not present in Laboratory Analysis Dataset (ADLB) – in order to represent the data from these subjects who did not receive treatment assignment, terminated participation in the study mid-way, lost to follow up, or screen failed.

For instance, the programmer decided to use open coding to better understand the concept about what needs to be displayed, including reaching out to the project statistician to develop a precise understanding that only subjects having a baseline record but missing post-baseline at the specific visits needed to be displayed regardless of their abnormal status. This was helpful for future programming practices with different tables as well as validating Statistical Analysis Plans and table shells.

If the programmer had simply downloaded the standard table template program, the entire process would have been faster with no fallouts, yet no improvement of the programmer's concept. This would just lead to the development of a fast workforce who would eventually be unable to develop further tables created based on concepts beyond the standard one. Discussion between programmers and statisticians to clarify what really needs to be displayed is crucial to producing the right program code. With increasingly desired shorter timelines by a smaller team, teams must mitigate the tendency to run standardized template programs without thinking critically and asking appropriate questions. In fact, a good deal of thoughts, ideas and confusions would not even form if the programmer did not first actively look at the table shell and think about how many display options may exist and how interpretations may differ, and where using standardized code versus new coding may be optimal.

- **TRAINING REQUIRED**

  Standardized macro templates are generated within industries by their own technical teams. Hence, these set of macros are not common between companies or industries. Training is required for new programmers as these templates are company or industry specific and there is usually a learning curve once a new programmer joins their team.

SUGGESTED APPROACH

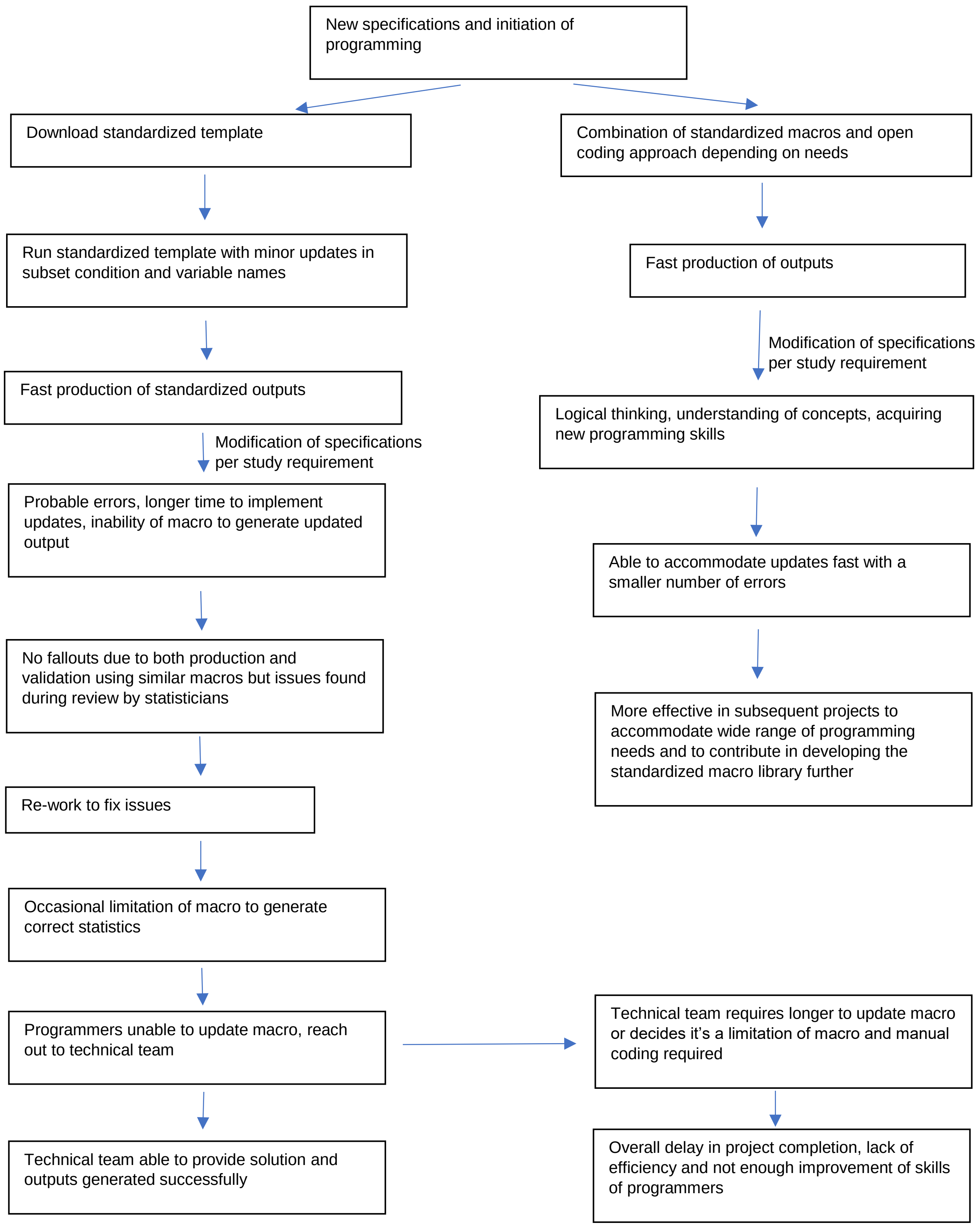
For standardized outputs, standardized macros are useful. But once the specifications are modified to be different from standardized parameters, open coding approach can be a better option. If the macro cannot be easily adapted to fit the requirement, it may be time to proceed with non-standard manual coding. Sometimes the use of macros may appear time-consuming in the hands of programmers other than the developer of the macros. In such scenarios, the best option may be the utilization of either standard or non-standard techniques depending on whether the programmer is comfortable with meeting the timeline. Also, a combination of both techniques helps develop better programming skills and concepts. Production and validation programs should use different macros to ensure true independence of the process.

With the increasing volume of work and short timelines for programming projects, using standardized macros can be favorable, with a balance between both standardized macro and open coding approach. Either or both approaches may be followed depending on the needs of specific studies. It is important that macros be robust enough to accommodate probable changes in outputs and validated beforehand. Proper documentation clarifying the purpose of each call needs to be in place.

Equally important is that programmers be able to see the codes running behind macro calls so that they can make necessary updates effectively and contribute to the development of the standardized macro libraries over time. Their access may be limited to read-only rather than full access to avoid unintended changes to the standardized macros. However, allowing programmers to access standardized macros and to use their open coding skills may yield higher productivity and better results for both the industry and the programmers. The best practice will not only depend on the reliability, reusability, and productivity of the standardized macros, but also on their readability, flexibility and extensibility along with a combination of traditional open coding approach.

A flow chart is presented below displaying the different outcomes of both approaches.

Diagram 1: Flow chart displaying advantages and challenges of different programming approach


New specifications and initiation of programming
Download standardized template
Run standardized template with minor updates in subset condition and variable names
Fast production of standardized outputs
Modification of specifications per study requirement
Probable errors, longer time to implement updates, inability of macro to generate updated output
No fallouts due to both production and validation using similar macros but issues found during review by statisticians
Re-work to fix issues
Occasional limitation of macro to generate correct statistics
Programmers unable to update macro, reach out to technical team
Technical team able to provide solution and outputs generated successfully
Combination of standardized macros and open coding approach depending on needs
Fast production of outputs
Modification of specifications per study requirement
Logical thinking, understanding of concepts, acquiring new programming skills
Able to accommodate updates fast with a smaller number of errors
More effective in subsequent projects to accommodate wide range of programming needs and to contribute in developing the standardized macro library further
Technical team requires longer to update macro or decides it’s a limitation of macro and manual coding required
Overall delay in project completion, lack of efficiency and not enough improvement of skills of programmers

CONCLUSION

Given the nature of work and short timelines of projects with the extensive programming needs, a suitable balance between the use of standardized SAS® macro templates and open coding approach is desirable over the stringent use of solely standardized macro templates. For the best quality outcome and highest level of efficiency, it is imperative to combine both approaches depending on the needs of a study.

REFERENCES

- Smoak, Carey. 2015. “Managing the Evolution of SAS® Programming” *Proceedings of the Pharmaceutical Industry SAS® Users Group 2015 Conference.* Orlando, Florida. Available at: https://www.pharmasug.org/proceedings/2015/CP/PharmaSUG-2015-CP03.pdf
- James, Margaret; Maass, Carolyn; Redner, Ginger. 2006. “Developing and Managing a SAS® Macro Library” *Proceedings of the NorthEast SAS® Users Group 2006 Conference.* Philadelphia, Pennsylvania. Available at: https://www.lexjansen.com/nesug/nesug06/po/po02.pdf
- Mo, Albert. 2006. “Developing, Managing and Evaluating a SAS® Macro System” *Proceedings of the SAS® Users Group International 2006 Conference.* San Francisco, California. Available at: https://support.sas.com/resources/papers/proceedings/proceedings/sugi31/018-31.pdf

CONTACT INFORMATION

Your comments and queries are valued and encouraged. Please contact the author at:

Tabassum Ambia
Alnylam Pharmaceuticals, Inc.
101 Main Street
Cambridge, MA 02142
Work Phone: 617-682-4294
Email: tambia@alnylam.com or ambia@bu.edu