Paper SM17

Towards Managing Macro Libraries in an SCE: A Vision for Reproducibility Through Version Locking

Shilpa Sood, Sycamore Informatics, India
Sanyogeeta Andhale, Sycamore Informatics, India
Priyanka Dharuvuri, Sycamore Informatics, USA

ABSTRACT

Macro libraries are foundational to modern statistical programming. This session explores a forward-looking approach to managing macro libraries within a Statistical Computing Environment (SCE), with the goal of improving access consistency, enhancing traceability, and reducing compliance risk. By envisioning a framework that incorporates modular release practices and version control directly into macro execution, organizations could enable systems where programs automatically capture and reuse the exact macro versions used at runtime—even if newer versions exist in the global library. Though not yet standard practice, this concept presents an opportunity to strengthen reproducibility, streamline deliverables, and align macro management with regulatory traceability expectations through future SCE capabilities.

INTRODUCTION

SAS macros have been the backbone of the statistical analysis and reporting process for several decades and it goes without saying that organizations, big and small, have one or more macro libraries to support their programming activities. Apart from utility macros such as the commonly used log check macro and macro to create bundles, organizations also have macros used at various stages of the programming process such as SDTM-specific macros, variable derivation macros for ADaMs and so on. Automation using macros may not be able to give teams bragging rights in this AI-age but it is hard to deny the time and hence cost saving that macros bring to our day-to-day programming deliverables across the industry.

From on operational perspective, there are 2 distinct aspects to macro libraries that organizations need to address. The first aspect is how programmers access a set of SAS macros that are available globally in their programming environment. The second aspect is how these macros are developed and tested, and when they are updated for bug fixes or enhanced functionalities, how they are released for the programming teams to use in their studies.

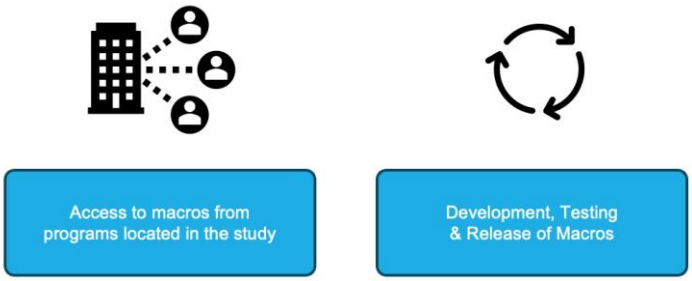

Figure 1: The 2 main operational aspects of macro libraries

Over the years, access to macros has been well-established across organizations and has been the topic of discussion for many a paper in industry-level conferences. There are 2 common approaches – either the programs in the study directly refer to macros in the global library or the macros are copied over from the global library to the study (or compound) at the start of programming activities. It is not uncommon to find organizations using a hybrid approach by capitalizing on the use of the SASAUTOS search path hierarchy to call macros located in multiple libraries at various levels in the search hierarchy.

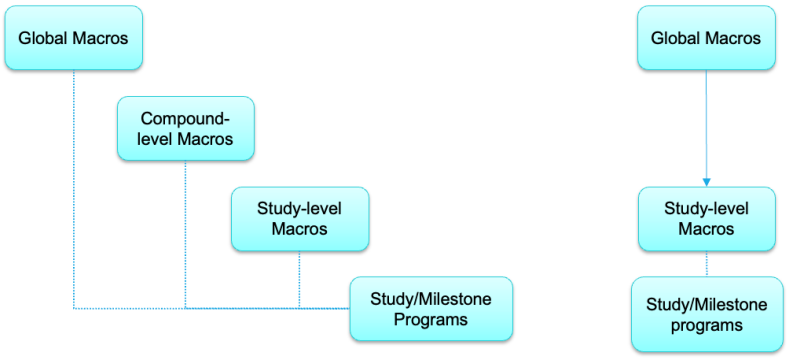

Figure 2: The 2 possible models of accessing macros for study programming activities

The macro release process followed by the systems/macro development in organizations is straightforward. In the absence of in-built version control and traceability aids, study programs always point to the latest macros and this works seamlessly until the study needs to be locked. However, as new regulatory requirements emphasize **traceability** and **reproducibility**, an additional layer is added in terms of managing these versioned libraries. Traditional methods, such as copying macros to individual studies or using separate folders for each macro release, are cumbersome and difficult to scale, especially with thousands of studies running concurrently. These legacy approaches can lead to decentralized macro versions, making bug fixes and updates a manual, error-prone process.

With the use of modern Statistical Computing Environments (SCEs) though, the regulatory requirements of versioning and traceability have been met, but a key challenge is the conflict between their built-in versioning capabilities and the need for reproducibility. When a macro is updated in the central library, programs that rely on an older version can become "stale" or "dirty" because the system flags the updated dependency. This impacts the reproducibility of a study, which is a major concern in terms of regulatory compliance. Although workarounds exist, such as maintaining a separate folder for each macro version, there is potential in technology to bring in novel solutions which could not be envisioned earlier. What can a modern SCE support to solve this seemingly trivial issue? Is there scope for the current technology to reduce overhead, improve automation and optimize existing programming processes resulting in smoother submissions and quicker time-to-market?

REGULATORY REQUIREMENTS AND MODERN SCES

Over the past decade, the term SCE has become commonplace in the industry. More than a decade ago, industry leaders and key stakeholders across organizations started to see the potential benefit that SCEs could provide. This resulted in significant advancements such as the creation of the SCE White Paper and the growth of vendors creating SCEs and SCE-like platforms. Not surprisingly, the programmers have been slower to react and adopt the newer systems since it calls for using newer interfaces, modified processes, and/or new techniques of programming. However, newer regulatory requirements **demand** the shift to GxP-compliant systems, and it is very hard to deny the amount of potential time and cost saving that teams and organizations can get in the long run by using smarter SCEs. Apart from simplified user management and rock-solid access controls, modern and smarter SCEs offer advanced programming aids such as in-built versioning, end-to-end data traceability and robust reproducibility. Complex programming workflows such as double programming and multi-tier review processes are very easy to track on such systems. The use of technology has made detailed, immutable audit logs a default in such systems and by nature, such systems can be easily integrated with existing third-party applications. Suffice it to say there is no end to the ability of such systems to provide surprisingly novel solutions to age-old problems plaguing the industry. What are some of the ways such modern applications can support seamless macro access to ensure fully reproducible analyses?

Figure 3: Modern SCEs cater to all the latest regulatory requirements

CHALLENGES AND OPPORTUNITIES

When organizations migrate their statistical programming activities from legacy systems to modern GxP-compliant, technology-enabled SCEs, what seems like a insurmountable task and fraught with resistance from multiple stakeholders can be perceived to be abundant in opportunities for organizations to optimize or even re-think age-old processes. Processes can be related to the creation of a folder hierarchy for a new study, controlling access to the study areas, restricting access to unblinded data or how to ensure seamless access to newly released macros. Working with systems that offer in-built version control and advanced traceability features gives us an opportunity to re-think study-level macro access for programmers and possibly even the macro develop-test-release cycle.

Let us now take a closer look at how modern SCEs achieve traceability using an example. The figure below shows how one or more programs located in a study accesses one or more macros located in a global library. Programs in this example point to the latest version of the macro in the global library.

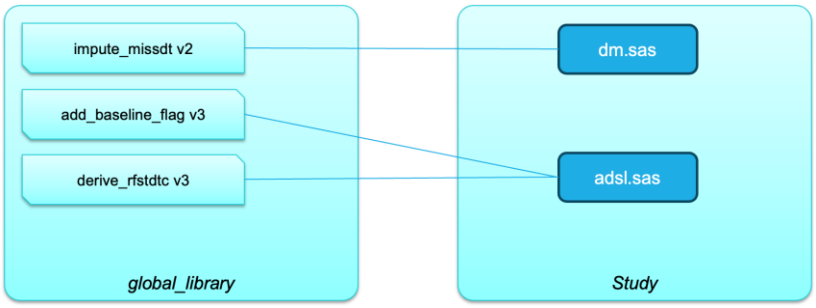

Figure 4: Programs establish traceability by pointing to the latest versioned macros

When one of the macros, say the `add_baseline_flag` macro, gets updated in the global library, all the programs that use this macro would display a "stale" or "dirty" symbol as shown below, nudging the programmer to re-run the program to get rid of the flag.

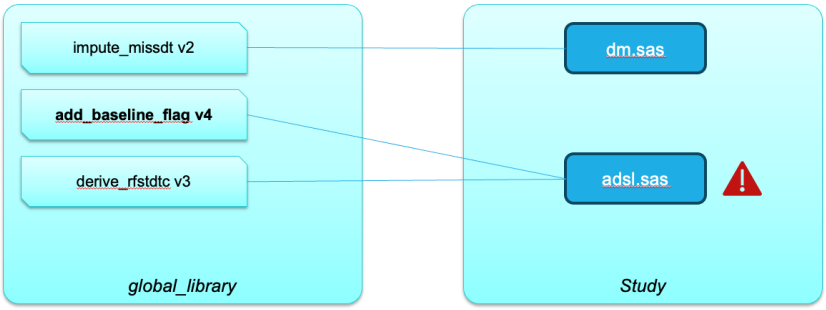

Figure 5: The extremely informative stale/dirty flag

The "stale" or "dirty" flag is extremely useful in all programming circumstances except when the study is locked, when it becomes undesirable. Flagging all the programs in a study can cause more than just unnecessary concern, since programs cannot be re-run to rid them of the flags after the study locked.

PROPOSED SOLUTIONS

There are several ways to deal with this scenario.

SOLUTION #1: The simplest way is to copy over the global macros to the study area prior to the locking the study. Until the study is locked, the programs get the full benefit of the latest global macros. This approach, however, adds an additional overhead to the study lead who must copy over the macros to the study macros folder, re-align the macro search paths and then re-run all the programs to ensure everything is working as expected.

SOLUTION #2: The second solution involves the macro development team taking a slightly different approach to handling their release process. Instead of updating the macros in a central location for every release, new and/or updated macros can be released into new folders that are organized by date as shown below. When a new release folder is created for an upcoming release, unchanged macros from the previous release are copied over to the new folder along with any new and/or modified macros that are part of the new release. None of the older release folders are modified when there is a new release. In this approach, when a new study is started, the team may wish to point to a specific release folder and stay with it over the duration of the study programming activities. At some point, they may also choose to switch to a newer release in case it offers a vital bug fix or enhanced functionality relevant to their study. In either case, there is no danger of a program turning stale due to a macro update after the study is locked.

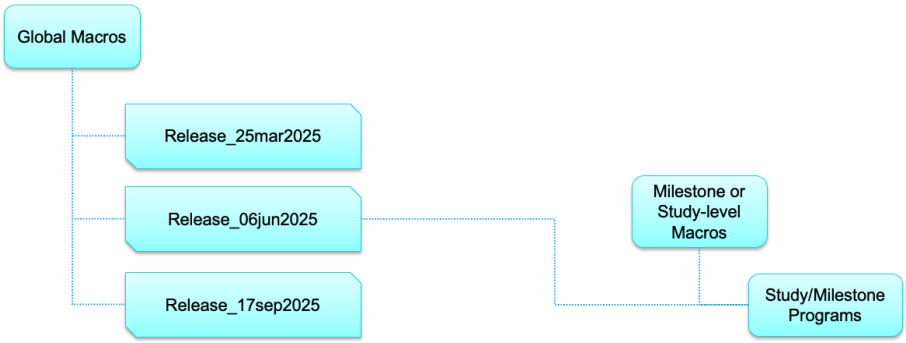

Figure 6: Macro releases in separate folders and a study refers to a specific release folder

SOLUTION #3: The third approach is to have study programs point to the macros in a centrally managed global library as in solution 1. The difference lies in the way programs access these macros. Instead of pointing to the latest version of the macros, the programs point to a version of each macro identified by a label, as shown in the figure below. This label can be specific to a delivery or can be specific to a release of the macro. This is a well-known methodology in the software development industry where a similar approach is used to put together software builds for every new release.

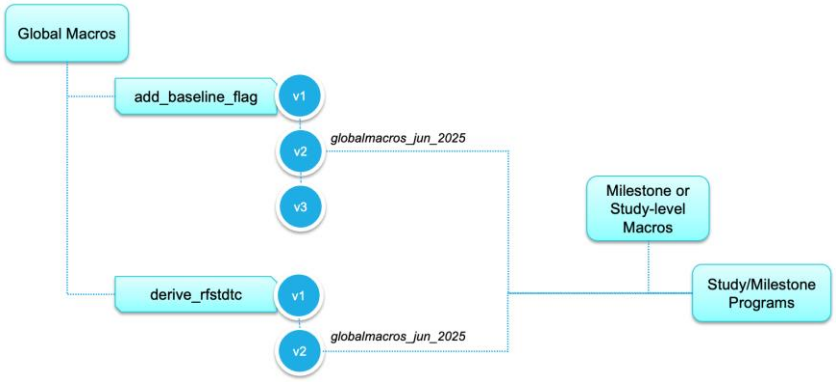

Figure 7: Study programs point to macros using a label

In this example, the study programs point to version 2 of the `add_baseline_flag` macro in the global library even though there exists a newer version of the macro that was possibly created after the study programming was started. A new label can be created for every new release from the systems development team and applied to the latest version of the macro as shown below. In cases where a macro undergoes no update in a particular release as in the case of the `derive_rfstdtc` macro shown in the figure below, the new label will also be applied on the older version of the macro where the older label or labels are applied. New labels, once applied, can then be sent out in the new release announcement so that study leads and programmers can choose to refer to the newer release.

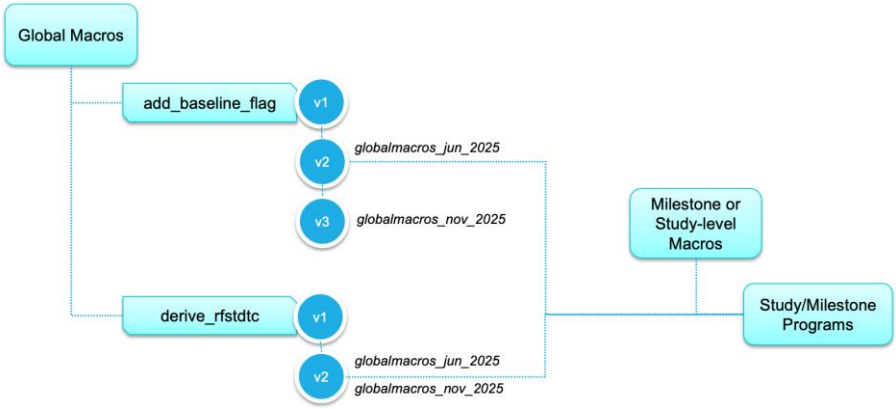

Figure 8: A new label is applied for all global macros for every new release

This approach of labelling files can be extended to cover not only global and/or study-level macros, but also raw datasets and supporting files such as lab parameter files and specifications. This allows all the files associated with a single deliverable located in a study to be tagged with a specific label as shown below.

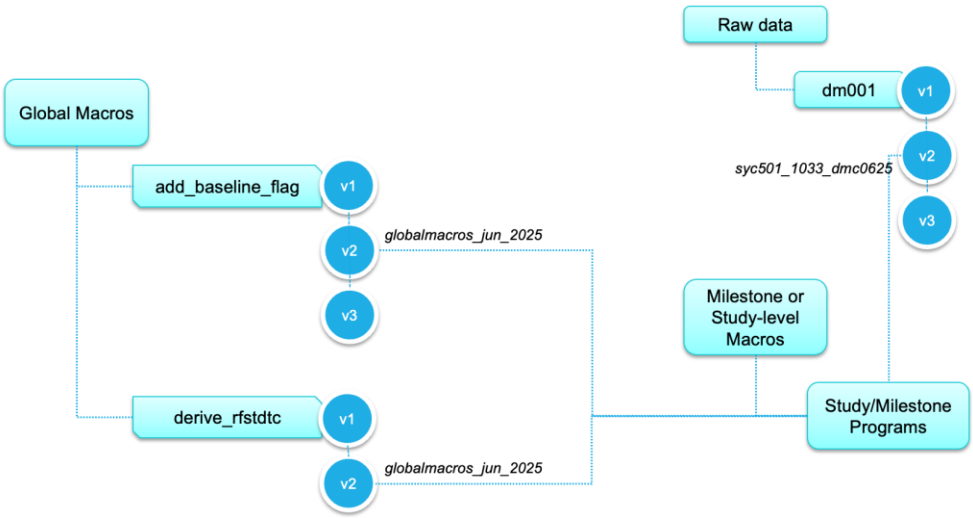

Figure 9: The use of labels extended to files located in a study

Extending this further can enable study programming teams to work with a single set of versioned programs in a study to manage multiple milestones. In the figure below, there are 2 sets of labels applied on the study programs and re-running older deliverables will only require the programmers to choose a specific label and re-run all the programs using that label. The system can also support using no label, in which case it will consider the latest version of the programs, data and macros.

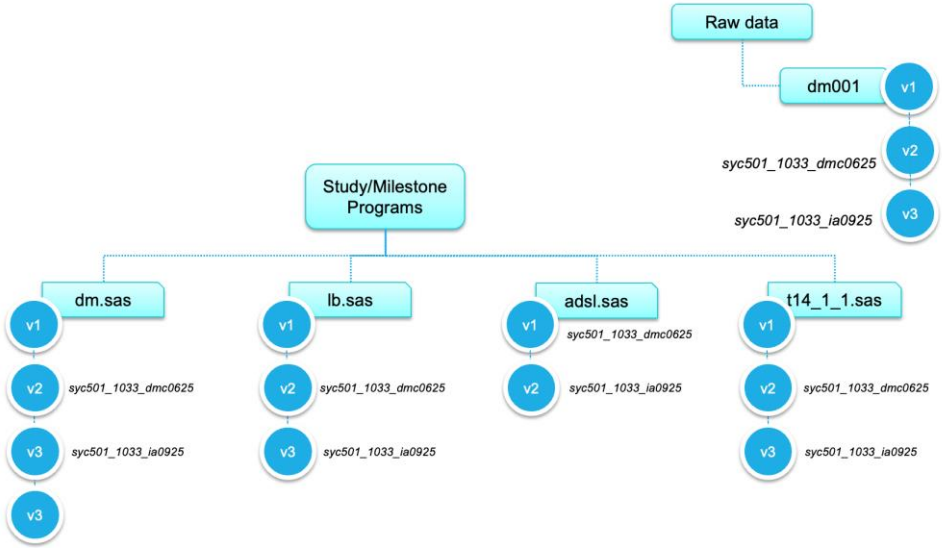

Figure 10: The use of labels can revolutionize the way we do our programming deliveries

Ensuring that each program, its inputs, and possibly even its outputs have labels can ensure accurate end-to-end traceability and reproducibility.

CONCLUSION

This paper proposes a forward-looking solution by envisioning a framework where macro libraries are managed centrally and programs can be "locked" to a specific macro version when needed. During development, a program can use the latest macro version available. However, at a critical point—such as a database lock or before a deliverable—the program can be labeled to lock in the exact macro versions used at that moment. This approach ensures that a program's behavior remains reproducible, even as new macro versions are released. Our vision aligns macro management with modern software engineering practices, strengthening reproducibility and streamlining deliverables to meet regulatory expectations. This approach not only enables accurate end-to-end traceability and reproducibility but also has the potential to revolutionize the way we do our regular programming deliveries. It has the

potential to reduce duplicated code and redundant efforts and also keep all the study programs consistent and in one reliable location.

REFERENCES

Arthur L. Carpenter, Building and Using Macro Libraries, SUGI 2002, Paper 17-27.
Mark Bynens et. al., Statistical Computing Environment – White Paper, 2019-2021

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Author Name: Shilpa Sood
Company: Sycamore Informatics
Address: Bangalore, India
Email: Shilpa.sood@sycamoreinformatics.com
Website: www.sycamoreinformatics.com