Paper SM12

Seamless Reproducibility: Version-Aware Global Macro Invocation in Statistical Computing Environments

Piyusha Mathur, Sycamore Informatics, India
Shilpa Sood, Sycamore Informatics, India

ABSTRACT

Macro libraries are a cornerstone of statistical programming, enabling standardization, reuse, and automation across studies. This paper presents a forward-looking approach to macro management within modern Statistical Computing Environments (SCEs). The proposed concept introduces version-aware macro execution, where programs automatically retain and reuse the exact macro versions used at runtime, even when newer versions are released in a centralized library. Although not yet a standard capability of most SCEs, this approach has the potential to enhance reproducibility, strengthen traceability, and reduce compliance risk. By aligning macro management with proven software engineering principles, organizations can simplify study maintenance, streamline regulatory submissions, and improve overall confidence in analytical outputs.

INTRODUCTION

SAS macros have long been integral to statistical analysis and reporting workflows in the pharmaceutical and life sciences industry. Most organizations maintain one or more centralized macro libraries to support routine programming activities. These libraries typically include utility macros for tasks such as log review and output packaging, as well as domain-specific macros used for SDTM mapping, ADaM derivations, and table, listing, and figure (TLF) generation. While emerging technologies continue to reshape the analytics landscape, macro-based automation remains highly relevant. Macros reduce repetitive coding, promote consistency across deliverables, and deliver measurable time and cost savings. As a result, effective macro management continues to be a critical operational concern for statistical programming teams.

From a governance perspective, macro libraries introduce two closely related challenges. The first concerns how programmers access shared macros within a study environment. The second relates to how macros are developed, tested, released, and maintained over time, particularly when updates introduce defect fixes or enhanced functionality.

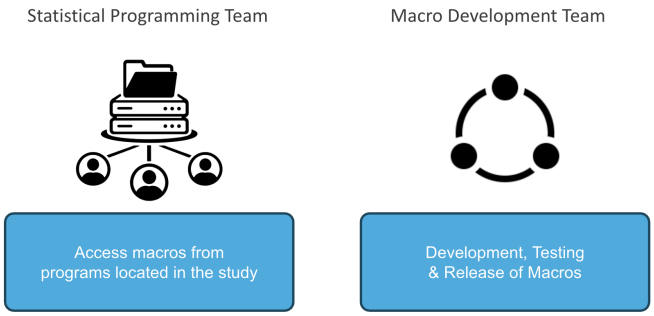

Figure 1: The 2 main operational aspects of macro libraries

Over the years, organizations have adopted a limited number of well-established approaches to provide macro access to study programmers.

1. **Direct global access**, where study programs reference macros directly from a centrally managed library.
2. **Study-local copies**, where macros are copied from the global repository into the study area at the start of programming.

Many organizations adopt a hybrid strategy that leverages the SASAUTOS search path to resolve macros from multiple locations in a predefined order. This allows teams to combine the convenience of global libraries with the stability of study-specific overrides.

Although these models are operationally effective, they become increasingly complex when regulatory requirements for traceability and reproducibility are introduced. In traditional environments, study programs typically resolve macros to the most recent version available, which can create challenges once a study approaches database lock or submission.

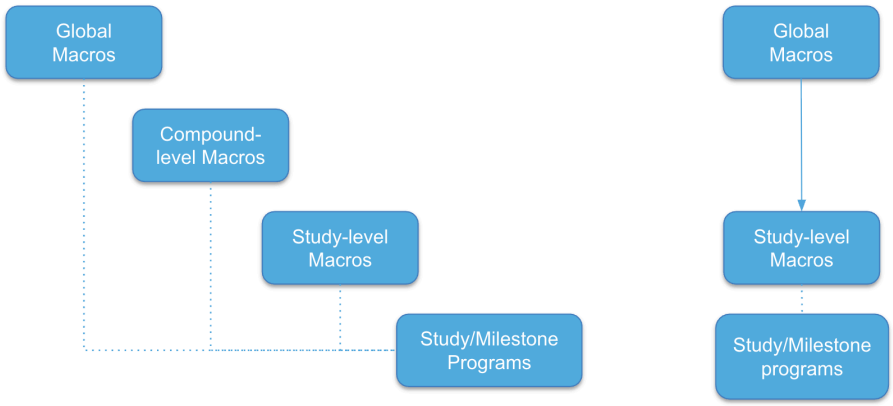

Figure 2: The 2 possible models of accessing macros for study programming activities

In many organizations, the macro release process managed by systems or macro development teams is relatively simple. Traditional practices—such as copying macros into individual study areas or maintaining separate folders for each macro release—can become cumbersome and difficult to scale, particularly when thousands of studies are running in parallel. These legacy approaches may also result in decentralized macro versions, making defect fixes and enhancements more manual and potentially error-prone.

Modern Statistical Computing Environments (SCEs) have significantly advanced the ability to meet regulatory expectations for version control and traceability. However, they also introduce a nuanced challenge: a tension between built-in versioning mechanisms and the need to preserve reproducibility. When a macro in a centralized library is updated, programs that depend on an earlier version may be flagged as out-of-sync or "stale," even if they were validated against the previous release. This can compromise the reproducibility of study outputs—an essential requirement for regulatory compliance. While interim approaches exist, such as maintaining dedicated folders for each macro version, these solutions often reintroduce manual effort and structural complexity. Advances in technology now present an opportunity to rethink this paradigm and explore approaches that were not previously feasible. What capabilities could a modern SCE offer to address this seemingly simple yet impactful challenge? And how might current innovations help reduce operational overhead, enhance automation, and streamline programming workflows—ultimately enabling smoother submissions and faster time-to-market?

The challenge for modern SCEs lies in balancing the benefits of macro-level management with the requirement for locked studies to stay fully reproducible.

Regulatory Requirements And Modern SCEs

Over the past decade, the term Statistical Computing Environment (SCE) has become well established within the industry. Even earlier, industry leaders and key stakeholders began to recognize the potential value that SCEs could deliver, leading to important milestones such as the development of the SCE White Paper and the emergence of multiple vendors offering SCE and SCE-like platforms. Adoption among programmers, however, has progressed more gradually, as these systems often require a shift to new interfaces, revised processes, and alternative programming approaches. At the same time, evolving regulatory expectations increasingly mandate the use of GxP-compliant environments, making this transition difficult to avoid. In parallel, the long-term time and cost efficiencies offered by more intelligent SCEs are becoming increasingly evident at both team and organizational levels. Beyond streamlined user management and robust access controls, modern SCEs provide advanced programming capabilities, including in-built version control, end-to-end data traceability, and strong support for reproducibility. Complex workflows—such as double programming and multi-layered review processes—can be readily monitored and managed within these platforms. The underlying technology also enables detailed, immutable audit trails by design and allows for straightforward integration with existing third-party tools. Collectively, these capabilities highlight the potential of modern SCEs to offer innovative solutions to long-standing industry challenges. This leads to an important question: how can such platforms further enable seamless macro access while ensuring fully reproducible analyses?

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

Challenges And Opportunities

As statistical programming in modern, GxP-compliant SCEs increasingly rely on centrally managed macro libraries, updates to shared macros can cause dependent study programs to be flagged as no longer fully aligned with the current library state. During active development, such indicators are often useful, as they prompt re-execution and verification of outputs following macro updates. However, the significance of these dependency alerts changes once a study has been locked. In post-lock scenarios, where program re-execution is typically restricted, system-generated warnings may raise questions during audits or inspections, even when the original executions were appropriately reviewed, approved, and compliant at the time of delivery. This highlights the need to carefully consider how centralized macro management, dependency tracking, and long-term study reproducibility can be effectively balanced in contemporary SCE implementations.

To explore this further, let us examine how modern SCEs support traceability through a simple example. The figure below illustrates one or more study-level programs accessing macros stored in a global library. In this scenario, the programs reference the latest available version of each macro in the shared library.

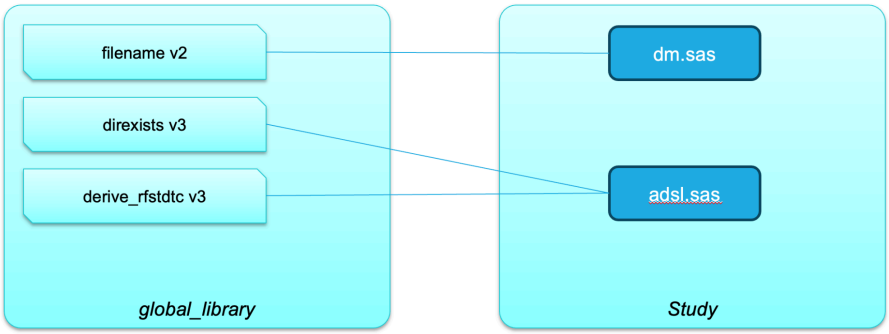

Figure 4: Programs establish traceability by pointing to the latest versioned macros

When one of the macros, say the `direxists` 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.

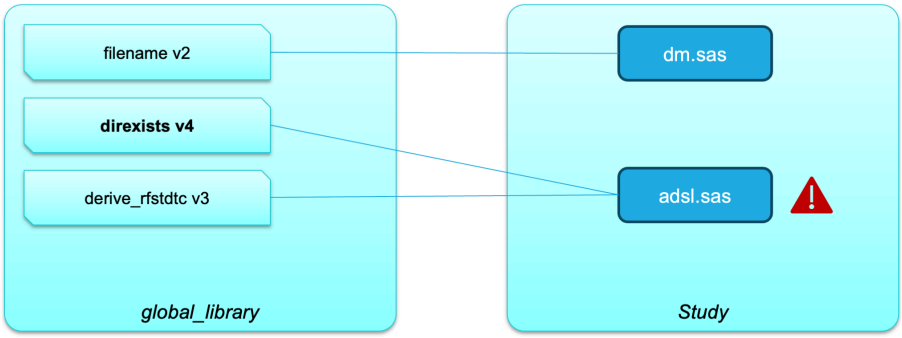

Figure 5: The extremely informative stale/dirty flag

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 is locked.

Proposed Solutions

There are several ways to deal with this scenario.

Solution #1: A straightforward approach is to copy the global macros into the study area before the study is locked. Until lock, programs can continue to benefit from the latest versions of the global macros. However, this approach introduces additional overhead for the study lead, who is required to copy the macros into the study-level macros folder, reconfigure the macro search paths, and re-run all programs to confirm that everything functions as expected.

Solution #2: The second solution requires the macro development team to adopt a slightly different release strategy. Rather than updating macros in a single central location for each release, new and/or updated macros are published into date-based release folders, as illustrated below. For each new release, unchanged macros from the previous release are copied into the new folder, together with any new or modified macros included in that release. Older release folders remain unchanged. With this approach, when a new study begins, the team can reference a specific release folder and continue to use it throughout the study programming lifecycle. If needed, they may later transition to a newer release that provides a critical bug fix or enhanced functionality relevant to the study. In either scenario, there is no risk of programs becoming stale due to macro updates after the study is locked.

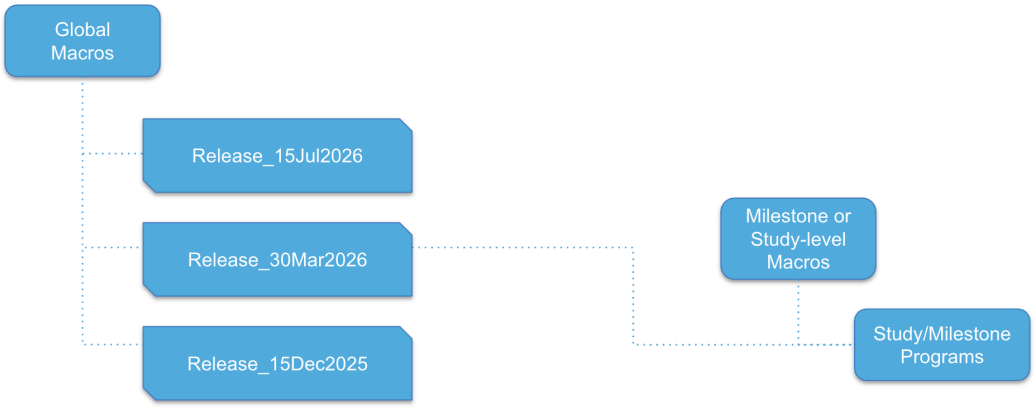

Figure 6: Macro releases in separate folders and a study refers to a specific release folder

Solution #3: The third approach also has study programs reference macros from a centrally managed global library, similar to the first solution. The key difference is in how the macros are accessed. Rather than pointing to the latest macro versions, programs reference specific versions identified by labels, as illustrated in the figure below. These labels may correspond to a particular delivery or a specific macro release. This approach mirrors a well-established practice in the software development industry, where labeled or tagged versions are used to assemble software builds for each release.

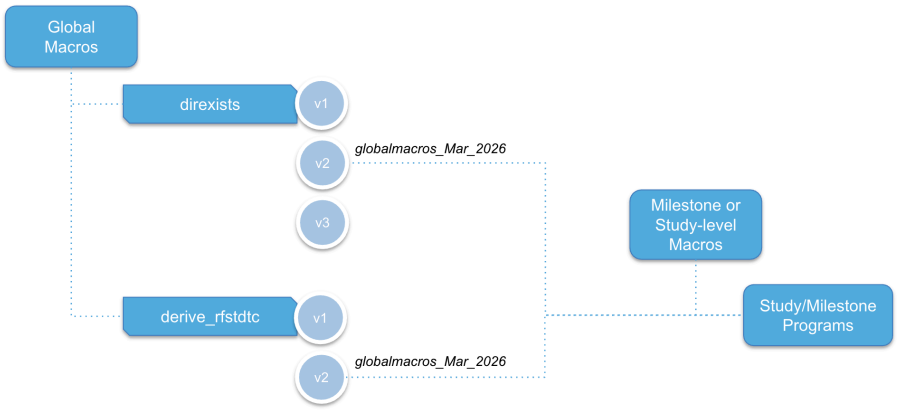

Figure 7: Study programs point to macros using a label

In this example, the study programs point to version 2 of the `direxists` 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.

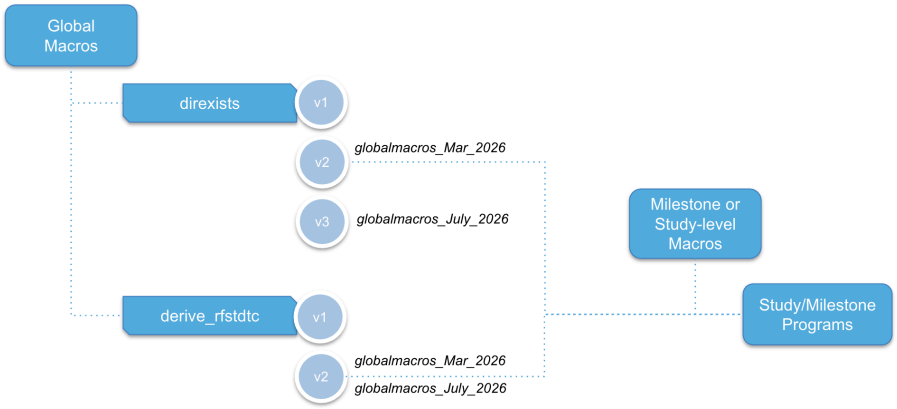

Figure 8: A new label is applied for all global macros for every new release

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.

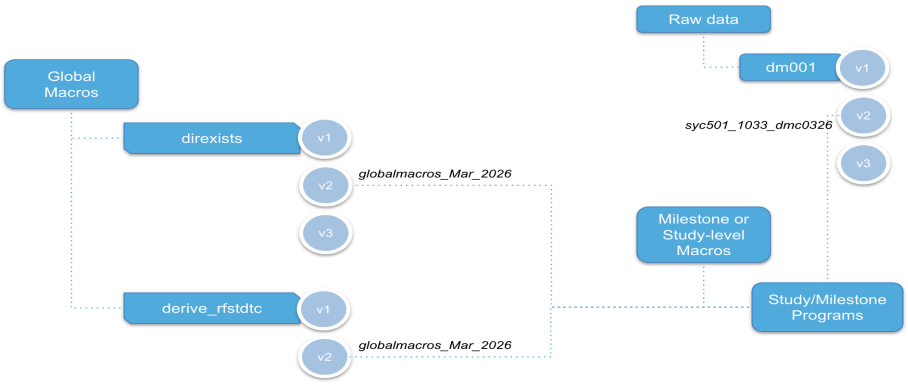

Figure 9: The use of labels extended to files located in a study

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 labels, in which case it will consider the latest version of the programs, data and macros.

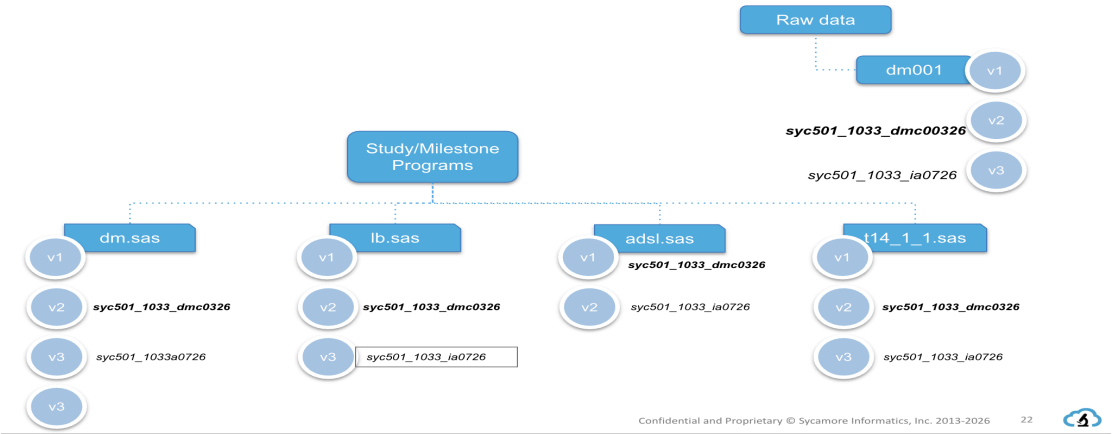

Figure 10: The use of labels can revolutionize the way we do our programming deliveries

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 more forward-looking approach by outlining a framework in which macro libraries are centrally managed, while programs can be “locked” to a specific macro version when required. During development, programs may continue to use the latest available macro versions. At a defined milestone—such as database lock or prior to a key deliverable—the program can be labeled to preserve the exact macro versions in use at that point in time. This ensures program behavior remains reproducible, even as new macro versions are subsequently released. This vision aligns macro management more closely with modern software engineering practices, supporting reproducibility and simplifying the preparation of regulatory-ready deliverables. Beyond enabling robust end-to-end traceability, the approach has the potential to reshape routine programming deliveries by reducing duplicated code and redundant effort, while keeping study programs consistent and maintained within a single, 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: Piyusha Mathur
Company: Sycamore Informatics
Address: Bangalore, India
Email: piyusha.mathur@sycamoreinformatics.com
Website: www.sycamoreinformatics.com