Paper SM04

Automated Programming Consistency Checks: Enhancing Clinical Programming Quality, Efficiency, and Scalability

Narendra Babu Kolisetty, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

Leveraging automation to improve accuracy and efficiency in clinical trial programming is vital for ensuring data integrity and regulatory compliance. This paper presents an innovative SAS macro-based solution that automates programming consistency checks by extracting and systematically analyzing macro parameter datasets across multiple programs. The utility macro applies predefined validation rules to identify programming issues such as missing ADaM and TLFs submission ready program files, incorrect subtitles, population and observation selections, therapy variables, AE percent incidence values in TLFs for DMC and CSR. Detected discrepancies are compiled into structured CSV files, with each tab corresponding to a specific check and containing detailed issue locations, affected output names, and explanatory notes to facilitate rapid resolution. Automating this process significantly reduces manual review time and human error. Designed for scalability and ease of use, this macro enhances programming quality, reproducibility, confidence in data analysis, supporting accelerated review cycles and successful regulatory submissions.

INTRODUCTION

Accurate and consistent programming is fundamental to the integrity of clinical trial data analysis. Statistical outputs such as tables, listings, and figures (TLFs) generated from analysis datasets provide the interpretable results for data monitoring committees (DMCs), clinical study reports (CSRs), and regulatory submissions. Maintaining programming consistency has become more challenging as clinical trials expand in scope and complexity. Clinical development programs often span across multiple disease areas and involve geographically dispersed programming teams working across time zones. In addition, analysis increasingly incorporates diverse data sources, including traditional clinical databases and external inputs such as genomic or biomarker datasets. While this integration enables comprehensive insights that support robust analysis and informed decision-making. It also heightens the risk of inconsistencies when program logic or macro parameter vary across deliverables.

Traditional manual review practices like program-by-program checks guided by ad hoc checklists are labor intensive, time consuming, and prone to oversight. These limitations are particularly evident when evaluating recurring macro calls, dynamic parameterization, or conditional preprocessing embedded within call programs. Undetected inconsistencies can lead to rework, timeline delays, regulatory questions, and diminished confidence in deliverables.

To address these challenges, this paper presents an automated SAS macro–based solution that systematically extracts and analyzes macro-parameter datasets to perform comprehensive programming consistency checks at scale. The solution is designed to detect issues earlier in the program development lifecycle, reduce manual review burden and generate structured, reviewer friendly outputs that facilitate efficient remediation.

MACRO ARCHITECTURE AND IMPLEMENTATION

The macro begins processing by parsing macro-parameter datasets produced when analysis datasets and TLF macro call programs are executed in SAS, thereby constructing a normalized input model for consistency checks. Validation rules are implemented as independent DATA step routines that operate on these input datasets to identify programming discrepancies.

Programming consistency checks are organized into several complimentary categories:

- **Presence checks** verify the existence of expected artifacts, such as submission-ready program files and macro-parameter datasets. Expected artifacts are derived from repository indices against the input macro-parameter records, and flagging any missing files or datasets associated with analysis datasets and TLFs for further investigation.

- **Consistency checks** compare macro-parameter values used in call programs including population selection, therapy-variable usage, and subtitle text and flag any contextual mismatches.

- **Syntactic detections** scan macro call programs to identify those that contain a preprocessing concept such as sas data steps, proc sort, transpose, merging, or deriving a new variable executed prior to macro calls, and flag them for additional review.

- **Semantic validations** assess whether analysis variables used in macro programs conform to study programming specifications and analysis dataset metadata.

- **Content checks** compare the parameter values specified in macro call programs with the values shown in TLF titles and other output content, report any discrepancies, and provide corrective recommendations.

All findings are consolidated into structured, tabbed excel workbook. Each worksheet is organized by issue type, providing a single, structured location for review and remediation. Results for each check follow a standard format that includes the affected output name, macro name, resolved macro-parameter values, expected value, and recommended programming remediation actions. An index worksheet summarizes all checks to facilitate efficient review and triage.

Figure 1 illustrates the SAS macro program flow, and Figure 2 provides the sample macro call program.

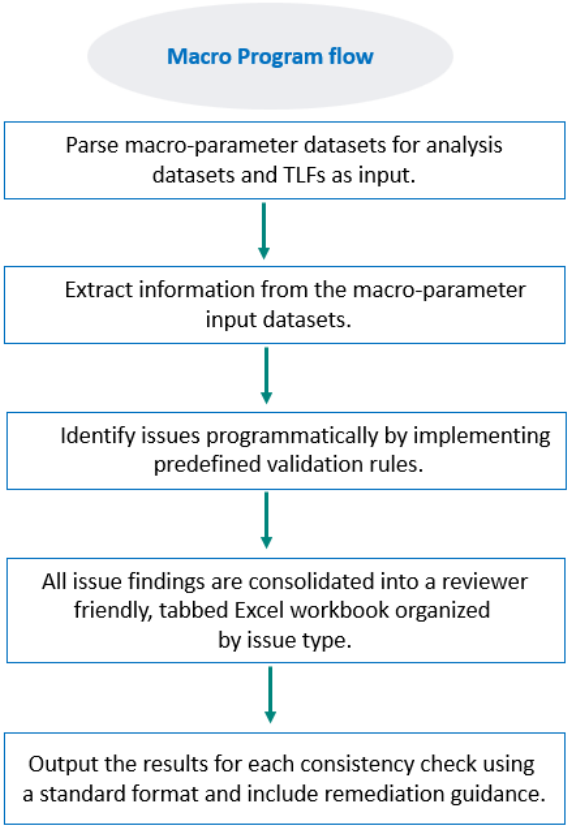

Figure 1. SAS Macro program flow

```
%check0prog (
input_library =  LPTIN                    /* input macro parameter datasets location */
,output_library = LPTOUT                  /* Path to output results*/
,output_fileref =  FPTOL                  /* fileref to output results*/
,outputs_select =  safety                 /* select required outputs to review */
,outpath = protpath/outlist               /* folder path to output an excel file*/
,rename_output=  prog0checks0&sysdate.    /* output filename */
,debug =   Y                              /* Y N ANALYSIS */
);
```

Figure 2. Sample macro call program

This macro is designed to process hundreds to thousands of programs across multiple studies with minimal manual intervention. Its modular design implements review checks as independent, maintainable modules, which simplifies extension, testing, and execution. The framework delivers deterministic and reproducible results supported by detailed logging and traceable metadata so that findings are repeatable. Usability is prioritized through reviewer-friendly, human-readable outputs (for example, tabbed Excel workbooks) that streamline QC review and triage. Finally, the utility is engineered for seamless integration into existing

program development lifecycles and operational workflows, including build processes and submission pipelines, facilitating adoption without disruptive changes to the established programming practices.

RESULTS

This macro program generates a consolidated Excel workbook that organizes all identified programming issues in a clear and consistent manner. Each issue type is exported to its own worksheet with descriptive tab names, as illustrated in Figure 3, allowing reviewers to focus on specific issue types without losing overall context.

To promote consistency, every worksheet follows a standardized column schema that includes fields such as the affected dataset or TLF program name, output name, resolved macro parameter values, expected value, explanatory note, and recommended remediation as illustrated in Figure 4. This uniform layout enables rapid navigation, supports consistent interpretation across checks, and reduces the time needed to reconcile discrepancies. Tabs are named descriptively for example, “Missing program files,” “Missing parameter datasets,” “Percent Incidence Mismatch,” and “Preprocessing step” so users can immediately understand the scope of each worksheet. In addition, each tab includes headers that allow reviewers to sort, group, filter, and prioritize high-severity items quickly. This helps teams identify the most critical issues first and track their resolution across review cycles.

Beyond the individual worksheets, the workbook also contains an index or summary tab, as shown in Figure 5, that aggregates key information across all checks. This consolidated view helps to link back to the detailed tabs, thereby streamlining navigation and facilitating cross-tab comparisons. By delivering results in this tabbed Excel format, the macro enables side-by-side examination of related findings and supports iterative triage by both programmers and reviewers. Consequently, teams can move seamlessly from identifying issues to assigning actions, monitoring progress, and confirming remediation, all within a single, coherent deliverable.

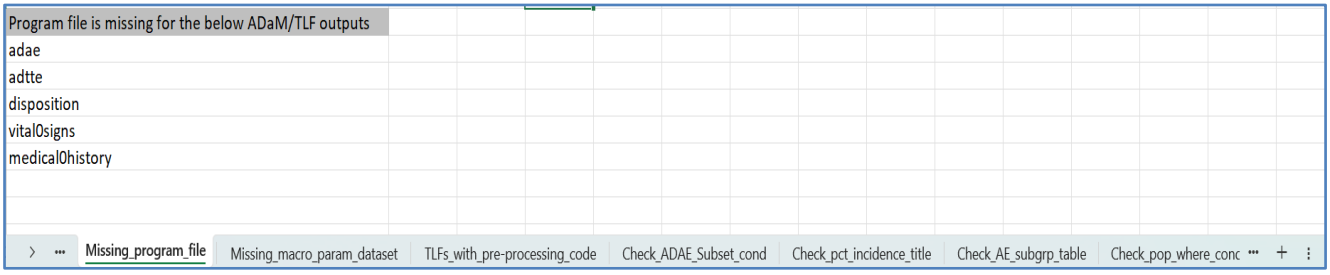

Figure 3 — Tabbed Excel workbook showing all checks and issue worksheets.

TLF Output Name	ADAE subset condition in the call program	Call Program Name	Expected Value/ Remidiation Guidance
ae0onset0gr35	SAFFL= 'Y' and TRTEMFL = 'Y' and PDEXCFL ^= 'Y' and atoxgrn in (3,4,5) and AOCC01FL='Y'	call0ae0time0to0onset	ACAT2 variable was not included in the subset condition
rainfall0all		call0ae0toxgr	Safety population subjects were not selected
rainfall0sae		call0ae0toxgr	All AEs were included. Need to keep only serious AEs.
ae0onset0dur	SAFFL= 'Y' and TRTEMFL = 'Y' and PDEXCFL ^= 'Y' and AEOSIGR1 ne " " and AEOSIFL="Y"	call0ae0time0to0onset	ACAT2 variable was not included in the subset condition
ae0outcome	SAFFL= 'Y' and TRTEMFL = 'Y' and PDEXCFL ^= 'Y' and AEOSIGR1 ne " " and AEOSIFL="Y"	call0ae0by0outcome	ACAT2 variable was not included in the subset condition

TLF Output Name	Call Program Name	Macro Parameter	Macro Parameter Value	Percent_Incidence_in_the_title	Expected Value	Explanatory Notes
specific0ae	call0specific0ae	percent_incidence	10		10	Percent incidence value is not matching between the value specified in the macro call program and TLF title
specific0ae0gr35	call0specific0ae	percent_incidence	1	0	1	Percent incidence value is not matching between the value specified in the macro call program and TLF title
toxgr0ae0pct5	call0ae0toxgr	percent_incidence	5	1	5	Percent incidence value is not matching between the value specified in the macro call program and TLF title

Figure 4 — Standardized output file structure with consistent column names and formats.

Sheet Name	Displays information about
Missing_program_file	Program file for the specified ADaM dataset and TLF outputs was not found.
Missing_macro_param_dataset	Macro parameter dataset for the specified TLF outputs was not found.
TLFs_with_pre-processing_code	Pre-processing code was found in the call program for the specified TLF outputs.
Check_ADAE_Subset_cond	ADAE subset condition in the TLF call program for the specified AE tables/listings is missing.
Check_pct_incidence_title	The percent incidence value displayed in the TLF title differs from the value entered in the call program for the specified TLF outputs.
Check_AE_subgrp_table	Checks ADAE variable used in the observation_where subset condition to generate AE sub-group tables for drug-related, gr3-5, action taken, SAE, and deaths.
Check_pop_where_cond	Actual treatment variable or Treated or Safety population flag was used in the call programs for the specified efficacy outputs. Planned treatment variable or Radomized or Allocated or Enroled flag was used in the call programs for the specified safety outputs.
Check_therapy_variable	Actual treatment variable was used in the call programs for the specified efficacy outputs. Planned treatment variable was used in the call programs for the specified safety outputs.
Check_subtitle	Subtitle does not match the population_where subset condition used in the call program for the specified TLF outputs.
Check_followup_table	Check the variables used to generate follow-up table for DMC/CSR.
Check_disposition_table	Check the disposition variables used from ADSL.
Check_prior_conmed_tables	An incorrect variable was used to subset ADCM records for the prior or concomitant medication reports.
All_Param	Macro parameter values used for all TLFs.

Figure 5. Index tab of the output workbook

BENEFITS & LIMITATIONS

Automating consistency checks for analysis datasets and TLFs programs brings several practical benefits. It removes repetitive, routine tasks from manual reviews workflows, allowing reviewers to spend less time on basic checks and more time on critical issues. Automation also speeds up review time and helps teams make decisions faster. Automation detects problems more reliably. Automated checks run the same way every time, so they can catch hidden inconsistencies and small mismatches that people often miss during manual reviews. Finding problems earlier in the program development cycle gives teams more time to fix them before they affect downstream workflows. Results become easier to reproduce because automated checks apply the same rules consistently, and the approach is scalable allowing the same checks to be run across larger datasets, more studies, or multiple programs with minimal effort.

The macro relies on having macro-parameter datasets available as input. When these parameter datasets are missing for a particular ADaM dataset or TLF output, the scope of consistency checks for those outputs will be limited. Minor updates to validation rules may be needed to reflect study-specific conventions for variable names and formats.

FUTURE ENHANCEMENTS

Planned enhancements include the development of a web-based dashboard using open-source tools such as R Shiny or Python frameworks. The dashboard will provide a centralized interface for issue review, triage, and track recurring problems over time. In addition, we also plan to add many more validation rules to catch wider range of inconsistencies.

Further improvements will focus on enhancing the macro parser to better handle complex and nested macro structures as well as integrating the framework with source control systems such as Git. This integration will support versioning, change tracking, and programmers and reviewers can collaborate more smoothly.

CONCLUSION

This work demonstrates that a SAS macro–based framework for extracting and analyzing macro-parameter datasets provides a practical and scalable solution for automating consistency checks in clinical programming. By programmatically checking parameter metadata by applying pre-defined validation rules, the framework reduces the manual burden on programmers and reviewers, surfaces errors earlier during code development, and strengthens program readiness for regulatory submission.

The modular design supports phased adoption and ongoing enhancement of the tool to address evolving clinical-programming practices and regulatory expectations without requiring a disruptive, all-at-once rollout. This tool increases operational efficiency and reduces regulatory risk by shortening remediation cycles and improving the traceability of fixes. This utility depends on the availability of macro-parameter datasets. Validation rules, which are based on baseline assumptions, must be tuned to accommodate study-specific conventions. Planned enhancements will further increase efficiency. Roadmap items include dashboarding for centralized issue management and trend analysis, expanding the rule library to increase coverage, improving parsing to handle more complex macro constructs, integrating with source control and continuous integration workflows. Together, these improvements will make the system more robust, easier to adopt, and more effective at prioritizing work. In summary, automated SAS macro-based consistency validation materially advances clinical-programming quality assurance by making checks more comprehensive, repeatable, and traceable. When implemented with appropriate governance and an incremental deployment strategy, the approach can substantially improve the reliability of statistical outputs and support timelier, more robust regulatory submissions.

ACKNOWLEDGMENTS

The author would like to thank Saigovind Chenna, Madhusudhan Reddy Papasani, Mary Varughese, and Amy Gillespie for reviewing the paper and providing their valuable feedback and inputs.

CONTACT INFORMATION

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

Author Name: Narendra Babu Kolisetty
Company: Merck & Co., Inc., Rahway, NJ, USA
E-mail: narendra.kolisetty@merck.com