Making Code Review Painless

Sy Truong, Meta-Xceed, Inc, Fremont, CA

ABSTRACT

Code review is an essential step in the development of concise and accurate SAS® programs. It is a required verification step in performing validation in a regulated environment. This paper will present techniques and a macro that can be freely downloaded to automate this task. The *%coderev* macro will perform many of the common tasks during a code review including:

1. Spell checking headers and comments
2. Reviewing all input and output datasets of the program
3. Comparing defined macro variables versus macro variable usage
4. Checking for multiple macro calls that are not in a macro library
5. Evaluating hard code logic
6. Evaluating sort order of all datasets

These tasks are normally performed by an independent reviewer instead of the original programmer. By automating the tasks, the code review process will ensure that the smallest mistake can be captured through reports to ensure the highest quality and integrity. What normally is a dreaded task can now be done with ease.

INTRODUCTION

In the larger scheme of SAS program verification, code review plays a significant role in ensuring the quality and integrity of your analysis. However, it is sometimes not viewed as important because code review is a mundane task that is not fully appreciated for the value it delivers. The verification of SAS programs and output include some of the following tasks.

Code Review	Systematic review of SAS programs according to a predetermined checklist of verification criteria.
Code Testing	Perform testing on SAS programs or macros supplying valid and invalid inputs and verify expected output.
Log Evaluation	Evaluate the SAS log for error, warning and other unexpected messages.
Output Review	Visual or programmatic review of report outputs as compared to expected results.
Data Review	Review attributes and contents of output data for accuracy and integrity.
Duplicate Programming	Independent programming to produce the same output for comparison.

This list includes verification tasks that may not apply to all programs since not all programs produce output analysis datasets or output reports. However, if you are verifying a macro that is used many times among all your team members, the time invested in performing as many of these tasks as possible is worth the effort. In a regulated environment, these tasks are not just recommendations but become requirements.

This paper will focus only on the first task of code review, although it will reference other aspects of SAS program verification. Even though the code review is the first item in the list of tasks, the order in which you perform your verification tasks can vary. They can be performed independently by different reviewers or by one reviewer either in parallel or serial. It is recommended that the reviewer is a different person from the original author of the SAS program to ensure objectivity. If possible, this effort can be outsourced to an external group so that there are no preconceived assumptions held from within your department. Once all the tasks are performed, a centralized summary report is compiled to capture all the findings. The process of reconciling these discrepancies goes beyond the scope of this paper but performing code review is an important step in this process.

CODE REVIEW CHECKLIST

Traditionally, code review was performed on paper. That was when programs and output were printed out and organized in a binder. The review process was then performed visually and a checklist was used to ensure that the reviewer did not overlook anything. An example of such a checklist looks like the following:

```
SAS PROGRAM
___ Program header complete and up-to-
    date.
___ Program commented clearly and
    adequately.
___ Macro variable definitions and
    usage are correct.
___ Data 'hardcode' corrections clearly
    commented with references in both
    the header and program.
___ Sorting orders correct.
___ Variables have been defined in the
    input datasets.
___ Macros used in more than one
    program are in the tools dir.

SAS LOG
___ Error check program clear.
___ Warning and error messages and
    warnings eliminated.
___ Un-initialized variables and merge
    notes eliminated.
___ Number of observations in each
    dataset as expected.
___ Checked all debugging messages
    (e.g. PUT statements).

SAS OUTPUT
___ All results have be validated
    either by outside methods (e.g.,
    proc freq or means, other) or
    against the CRT file (listing or
    SAS viewer).
___ Titles, footnotes and labels
    verified for accuracy and
    completeness.
___ Spelling is correct.  Checked for
    typos.
___ Labels for columns and rows are
    correct.
```

The checklist will vary with each group since there may be some departmental standards that are not universally applied. In general, however, the essence of the list is the same. The goal of the checklist is to ensure that independent reviewers do all the verification tasks required. This also ensures that the discrepancies found corresponding to the checklist are easily documented and understood across different programs.

Reviews are rarely done on paper any more since the process can be effectively done on screen. This allows the reviewer to use text searches or cut and paste between documents to more efficiently perform verification and documentation. Although the process may have changed, the construct of a list is still a good tool to have. No matter how high tech you get, it is still a good idea to know what it is that you are checking for while performing a code review.

AUTOMATING CODE REVIEW

Some critical analysis is performed during the code review process, but a large part of the task is repetitive. This is one of the main reasons why people dislike performing code reviews. After performing multiple iterations of the same review tasks, the work becomes mundane and the reviewer tends to get blurry eyed. This repetitive fatigue leads to sloppiness because the reviewer loses the fresh acuity of performing the very first verification task.

This presents an opportunity to automate the repetitive tasks to liberate the reviewer from performing the mundane tasks. This is the main objective for the use of the *%coderev* macro. This macro parses through the SAS program and captures things that are potentially discrepant as predetermined from a list review criteria. The macro then presents a report to the reviewer to help identify all items that are truly discrepant. *%coderev* has its own checklists of things it searches for. The list includes:

1. **Required Comment** - This will verify that there is a SAS comment preceding every data step and SAS PROC.
2. **Program Header** - This will verify if a comment header exists in the program.
3. **Input and Output** - This report will display all potential input and output produced by the program to assist in verifying against the program header.
4. **Spell Check Comments** - This captures all program comments for the purpose of performing spell checking. The report can be pasted into a word processor with a spell checker.
5. **Macro Usage** - This report presents all macro definition such as %let and call symputs. It will also capture all use of macro variables which begin with & so that comparisons can be made.
6. **Hard Code Logic** - This captures all potential hard coded logic. A review of such logic is recommended.
7. **Sort Order** - Verify all sort order of datasets to ensure proper usage.
8. **Macro Calls** - Verify any macro that is called more than once. This is to ensure that it should be defined in a standard tools directory rather than defined locally.
9. **User Defined Formats** - Capture any user defined formats that do not produce a catalog. This is a form of hard coding that is not recommended.
10. **Put Statements** - Capture all put or %put statements since this is usually intended for debugging. A review of the log is recommended against the put statements.
11. **Spell Check Titles and Footnotes** - All titles and footnotes are captured for the purpose of spell checking. The titles and footnotes are also compared for similarities. This is intended to verify for consistency.

In order to perform the list of verification tasks, the code review can be initiated through a macro call with the following syntax.

%coderev(path=path location of SAS programs, **program=**selected SAS program name with extension**);**

Where	Is Type...	And represents...
path	C (200)	Library referencing the location of the reports dataset.
program	C (200)	SAS program name. For example, *demog.sas*.

Upon request, the code review utility will parse through the specified SAS program and systematically look for conditions from the checklist. The macro is not intelligent enough to determine if the code is actually discrepant or not. Rather, it will present information in a summary report to help the reviewer perform the critical analysis and make the final decision.

The following is an example of an abbreviated report used to demonstrate checklist case number 3 and 5.

Code Review Report

Program Name	Case Number	Comments	Program Code or Log
A_ae.sas	3	Potential Input for Data Step	[34] set rawdata.ael;
A_ae.sas	3	Potential Input from SAS PROC	[69] proc sort data =anadata.ademog out=demo(keep=&keptvar);
a_ae.sas	3	Potential Output for Data Step	[73] data anadata.aae;
a_ae.sas	5	Macro variable created for: pgm	[13] %let pgm=a_ae;
a_ae.sas	5	Macro variable created for: keptvar	[15] %let keptvar=subjid usubjid studyid trtcd trtgrp std_dt1 mitt;
a_ae.sas	5	Macro Variable is Used.	[69] proc sort data =anadata.ademog out=demo(keep=&keptvar);

Generated on: 05/21/2004, 11:31:41 am, truong
Located at: /myserver/stat/study101/003/validation

In this example, the reviewer can see that the input datasets for this program included: rawdata.ael and anadata.ademog. The reviewer would then verify if the comments in the header and/or analysis plan match these results. For checklist case number 5, the reviewer can see that there were two macro variables defined: PGM and KEPTVAR. The report also shows that only KEPTVAR was used as a macro. This will prompt the reviewer to investigate why macro PGM was not used. The program code that is presented in the report is displayed with the line number in square brackets preceding each line. This also helps the reviewer quickly pinpoint the location of potential discrepancies.

This prompted investigation that occurs after reviewing the report is critical in identifying discrepancies. The algorithm sometimes does capture code that is discrepant but sometimes the code is fine. The report does not make this distinction but it allows the reviewer to quickly identify potential discrepancies and create those "a ha!" moments that might otherwise be overlooked.

GETTING %CODEREV

The *%coderev* macro can be freely downloaded for evaluation at: http://www.meta-x.com/syvalidate/codrev/. The code review is a macro which is part of a larger set of validation tools named Sy/Validate. Sy/Validate provides additional tools for keeping an audit trail and version control over your validation process. Results found from the *%coderev* can be stored and tracked so that you know the status of all your validation efforts.

CONCLUSION

There are many tasks performed in the process of verifying and validating SAS programs. Many of these tasks are overlooked for their significance in maintaining accuracy and integrity of the program logic and output which it produces. The repetitive

aspect of these tasks gives them a bad reputation of being unglamorize grunt work that must be done to meet regulatory or departmental SOPs.

Code review is one of such tasks that is not fully appreciated. It is an important step in a process that also includes: code testing, log evaluation, output review, data review, and duplicate programming. Code review can be optimized when accompanied by a checklist to ensure that all verification tasks are performed. Even though the list of tasks is clearly defined, the mundane aspects of some of the tasks can lead to sloppiness as the reviewer gets fatigued from doing it repetitively. The %coderev macro is design to address this issue by performing many of the mundane checks and summarizing the findings in a report. This report helps the reviewer perform the critical analysis and identify the discrepant code.

The goal is to retain the reviewer's objectivity during the review process even after reviewing hundreds of programs. The unbiased eye can also be maintained if the reviewer is independent from the programmer and ideally outside the group. This is an example where if the reviewer is outsourced as a third party, this can add fresh perspective that helps in identifying discrepant programming practices.

The %coderev macro does remove some of the mundane review aspects of the code review process, but the critical analysis done in identifying the deviant code is still needed by the reviewer. From a reviewer's perspective, the removal of the boring repetitive aspect of the work makes the process much less painful and leaves the challenging aspects of an audit. What is left is more challenging and fun.

REFERENCES

ABOUT THE AUTHOR

Sy Truong is a Systems Developer for Meta-Xceed, Inc. They may be contacted at:

Sy Truong

48501 Warm Springs Blvd. Ste 117

Fremont, CA 94539

(510) 226-1209

sy.truong@meta-x.com