Paper 69-2018

A Simple Method for Colorizing Saved Log Files

Matthew Slaughter, Kaiser Permanente Center for Health Research

ABSTRACT

This presentation discusses a method for reading SAS® logs saved as text files from previously executed SAS jobs and displaying them in the log window of a current SAS session. This allows SAS to color code error, warning, and note messages in saved logs which lose this formatting when saved as text. This enhances the readability of SAS logs for code review. Other methods for saving logs with color will be briefly discussed, as well as the advantages and disadvantages of those methods. The SAS log is an important tool for ensuring the quality of analytical work in SAS and color coding helps to highlight information provided about SAS code execution.

INTRODUCTION

The SAS log is a useful source of information about program execution, aiding programmers in the task of debugging and optimizing programs, as well as simply providing a historical record of the code that is run in a particular batch or interactive SAS session. The utility of the SAS log can even be extended by the addition of user-generated messages. While displayed in the log window of a SAS programming interface such as the SAS Windowing Environment, SAS Enterprise Guide, or SAS Studio, these messages will be helpfully color coded to increase readability and allow different types of messages to be easily identified by casual inspection. Organizations may expect or require the review of SAS logs to ensure the appropriate use and management of data sources, the accuracy of analytical results, and the optimal use of computing resources. For example the Kaiser Permanente Center for Health Research requires review of SAS logs whenever the results of SAS programs contribute to a published analysis, generate a dataset for transfer offsite, or generate patient contacts for the aforementioned reasons as well as to ensure appropriate exclusion checks have been applied and other relevant policies and best practices are followed.

However, it often impractical or impossible to review logs in the same interactive session in which code was run. Instead, saved log files resulting either from batch processing and/or with the use of the PRINTTO procedure.

```
filename mylog '<filepath>';
filename mylis '<filepath>';
proc printto log=mylog print=mylis new;
run;
```

Unfortunately, since log files saved in this way contain only plaintext with little in the way of formatting, the color coding that would be available in a SAS session log window is generally lost. While it is possible to use the -RTFCOLOR initialization option and the WRTFSAVE command in the SAS windowing environment to save a SAS log to an RTF file with color, this command is not available in other SAS interfaces nor in batch mode[1]. It is also possible to print the contents of a SAS log window to a PDF to preserve color coding, but unfortunately there is no way to do this programmatically. Nevertheless, it is possible to read the contents of a saved log file and print the contents to the log window of an interactive SAS session, allowing color-coding to be restored.

READING SAVED LOGS

It would be convenient if SAS would simply display opened log files in a log window, but given that the purpose of log window is to be record of code executed in a given interactive session, it's not surprising

[1] Note that -RTFCOLOR will not work with PROC PRINTTO; while this procedure can write to a file with a .rtf extension, the file will not contain any rich-text formatting.

this functionality does not exist. Fortunately, it is easy to read a log file into the low window of an interactive SAS session using the DATA step.

```
filename mylog '<filepath>';
data _null_;
    infile mylog;
    input;
    put _infile_;
run;
```

The above code reads the log file specified in the INFILE statement line by line and prints each line to the log of the current session using the PUT statement. The _INFILE_ automatic variable references the current input buffer (effectively the contents of each line of the log file in as they are read on each iteration of the data step). In order to allow users to create custom error messages, warnings, and notes, SAS automatically color-codes any line starting with the relevant keywords written to the log by a PUT statement or other programmatic methods. Thanks to this feature, the messages read from the saved log will be appropriately color coded when viewed in the log window of the current session after being read programmatically.

LIMITATIONS AND FORMATTING ISSUES

The downside to this method is that lines of the log which do not start with an appropriate keyword such as "ERROR:" or "NOTE:" will not be color coded. This means, for example, that while the message "NOTE: DATA statement used (Total process time):" will be recognized as a note and colored appropriately, the real time and CPU time statistics usually reported directly below will not, whereas they would have been if they were generated in the current SAS session.

Perhaps more problematically, messages that are broken up across two lines in the source log file will written to the log separately, with only the first part being correctly color coded. This can be avoided by using the LINESIZE system option to avoid unnecessary line breaks (to be really sure set it to MAX), but this does require planning ahead; it's the LINESIZE setting of the SAS session which generated the log file which matters, not session reading it.

```
options linesize max;
```

If the log file contains tabs for formatting, they may be stripped out when read by the data step. Adding the EXPANDTABS option to the INFILE statement helps to deal with this, and results in tabs being replaced with an appropriate number of spaces as determined by the settings of the current session.

```
data _null_;
    infile mylog expandtabs;
    input;
    put _infile_;
run;
```

An external log file also often contains page breaks. This isn't a serious problem if one only wants to view the saved log in the log window of the current session but attempting to print out a copy of such a log after restoring the coloring this way or saving it to a PDF will result unnecessary whitespace and a serious headache. One way to avoid this is to use the PAGESIZE option, similar to the way one specifies the LINESIZE option.

```
options pagesize max;
```

However, making the page size extremely large can have negative consequences, especially in a SAS program which generates listing output. As an alternative, consider adding a little conditional logic to program to avoid printing the lines which contain the page breaks.

```
data _null_;
    infile mylog expandtabs;
    input;
    if index(_infile_,'0c'x) = 0 and index(lag(_infile_),'0c'x) = 0
```

```
        then put _infile_;
run;
```

The INDEX function checks a string (in this case, the current line of the log being read) and checks for a character or string within it, returning a 1 if found and a 0 otherwise. '0C'x is a hexadecimal constant which indicates a page break character. Using the lag function, we can also check to see if the previous line contained a page break (SAS adds a header line to the top of each page which looks odd when it shows up at seemingly arbitrary points when printing the colorized log) and exclude such lines from printing to the log window. To do this you may wish to consider the use of the NOSOURCE system option. This excludes SAS source code from the current session from being written to the log. This will leave just the notes associated with the DATA step reading the log file, the contents of the log file, and the log line generated by submitting the OPTIONS statement itself. The notes associated with the DATA step could be eliminated by using options NONOTES, but they contain useful information about the log file being read, and, more importantly this would prevent notes read from the saved log file being printed to the current log window, which would be highly counter-productive. The better solution is to submit the OPTIONS statement, clear the log, and then submit the DATA step, leaving only the contents of the log to be colorized and useful information about the file from the DATA step.

```
filename mylog 'filepath';
options nosource;

/*Clear log here*/

data _null_;
    infile mylog expandtabs;
    input;
    if index(_infile_,'0c'x) = 0 and index(lag(_infile_),'0c'x) = 0
        then put _infile_;
run;

options source;
```

The specifics of how one may go about clearing the log vary somewhat depending on which SAS interface is used.

DIFFERENCES IN LOG CONTROL AND PRINTING BETWEEN INTERFACES

SAS WINDOWING ENVIRONMENT®

The SAS Windowing Environment (formerly known as Display Manager) log window maintains a record of all code submitted during a SAS session, unless cleared manually or programmatically. The latter option is notable, as the Windowing Environment is the only SAS interface which supports a programmatic means of clearing the log. This is done by means of the DM statement, which passes commands not to SAS itself, but rather to the Windowing Environment for the purpose of controlling the interface.

```
dm 'log;clear';
```

The downside of the DM statement is that the commands execute as soon as the statement is submitted—before the rest of the program compiles, no matter where the DM statement occurs in the program. To force it to execute midway through a program, it's necessary to wrap the DM statement inside regular SAS code, such as the CALL EXECUTE routine.

```
data _null_;
    call execute("dm 'log;clear';");
run;
```

Simply insert the above data step after the OPTIONS statement to create a log reading program which requires only the path of the log file to be edited before running once.

Having read the contents of the save log into the active log window, it's easy to print to either a hard copy or a PDF with color coded messages. Consider adding page numbers from under "Options..." in the print window, and check the line size and page size under "Setup...".

Finally, note that one may configure the appearance and color settings of the log window from the SASCOLOR window Found under tools -> options -> colors...

SAS ENTERPRISE GUIDE®

The Enterprise Guide program log provides a record of the last code submission from the associated SAS Program. The project log summary allows a quick view of messages in the log, including user generated ones and those read from a previously saved log file. Unlike the Windowing Environment log window, log messages from previous submissions are cleared each time code is run. This makes hiding the OPTIONS statement from the log as easy as running the program in two parts. However, EG does add wrapper code to the log, which is important to keep in mind. Wrapper code can be hidden from the log via the appropriate setting in Tools -> Results -> Results General, but using OPTIONS NOSOURCE will also hide it. The colors used for different types of log messages can be changed from Program -> Editor Options -> File type: SAS log file.

SAS STUDIO®

By default, the log tab in SAS studio operates similarly to the program log in Enterprise Guide. However, SAS studio has no feature to hide the wrapper code it uses in the log. It is, however possible to disable the wrapper code entirely by activating interactive mode. This feature makes the log tab operate like a hybrid of the Windowing Environment log window and an EG program log. Like the former, it records all code run across multiple submissions and needs to be cleared manually. Like the latter, it only records statements submitted from a particular program. I recommend using interactive mode and clearing the log manually after submitting the FILENAME and OPTIONS statements to produce a "clean" view of the saved log file.

Printing the log from SAS studio will open the log in a separate browser tab, at which point printing is handled by the settings of the browser.

Currently SAS studio does not provide the ability to change the colors used in the log, but this feature may be provided in a future release.

CONCLUSION

Reading a saved SAS log file using the DATA step provides a simple method for restoring the color coding of SAS errors, warnings, and notes. When a manual review of a SAS log is appropriate, color-coding these messages can improve legibility to allow better and more reliable evaluation of efficiency, quality, and compliance. This paper only discusses the most common SAS interfaces, but any SAS interface which supports the display of color-coded log messages would likely support a similar method for displaying a saved log file in color. Treating a log file as a data source opens up a lot of possibilities for parsing the program metadata contained within, but valuable enhancements to the usability of a saved log can be made without the need for complicated or advanced programming.

REFERENCES

Delwiche, Lora D. & Slaughter, Susan J. 2017. *The Little SAS Enterprise Guide Book*. Cary, NC : SAS Institute.

T J. 2008. “Opening SAS Logs with color coding (.log files)” Accessed July 30, 2018. https://groups.google.com/d/msg/comp.soft-sys.sas/SkTR6bmBSEk/l_uyApk4qYoJ.

CONTACT INFORMATION

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

Matthew Slaughter
Kaiser Permanente Center for Health Research
Matthew.T.Slaughter@kpchr.org

FULL CODE

```
filename mylog '<filepath>';
options nosource;

*Skip this step if not using SAS EG or SAS Studio;
data _null_;
    call execute("dm 'log;clear';");
run;

data _null_;
    infile mylog expandtabs;
    input;
    if index(_infile_,'0c'x) = 0 and index(lag(_infile_),'0c'x) = 0
        then put _infile_;
run;

options source;
```