Paper SY05

Cutting the SAS® LOG down to size

Malachy J. Foley, University of North Carolina at Chapel Hill, NC

ABSTRACT

Looking through a large SAS LOG (say 250 pages) for NOTE's and WARNING's that might indicate a problem with your SAS program is necessary, but no fun. This paper presents a SAS program which distills the SAS LOG to list only the critical messages in a LOG. The use of the program increases program reliability and programmer productivity.

INTRODUCTION

The SAS LOG is a wonderful tool for debugging the SAS program. It provides a listing of all your original code plus a myriad of messages that comment on your code. Good SAS programmers know that they need to thoroughly review the SAS LOG to find any message that might point to a bug in their code. However, pouring over endless pages of LOG is a time-consuming, tedious, and error-prone process.

This paper shows a program that will take the drudgery and missed message out of the LOG reviewing process. It even shows a second program that can automate many of the steps involved in running a SAS program in the Batch Mode.

By the end of this paper the reader should be able to implement programs and strategies that will simplify their life while programming in the Batch Mode.

A TYPCIAL SAS LOG

Exhibit 1 shows part of a SAS LOG for an example program which has some coding problems. This program is called BAD_CODE.SAS. The part of the LOG displayed in Exhibit 1 came from the file named BAD_CDOE.LOG

Exhibit 1: Example of a Part of a SAS LOG Which Contains Coding Difficulties

```
1          * Example code with problems;
2           optns ls=75;
            ______
            14
WARNING 14-169: Assuming the symbol OPTIONS was misspelled as optns.

3           data _null_;
4                X=0.0;
5                y=5/X;
6                length z $5 U 8;
7                z="qwert";
8                w=substr(z,50,5);
9           run;

NOTE: Variable U is uninitialized.
NOTE: Division by zero detected at line 5 column 9.
NOTE: Invalid second argument to function SUBSTR at line 8 column 8.
X=0 y=. z=qwert U=. w=qwert _ERROR_=1 _N_=1
NOTE: Mathematical operations could not be performed at the following
      places. The results of the operations have been set to missing
      values.
```

One of the nice things about the SAS LOG is that each line of code is prefixed with a line number and indented. On the other hand, all the messages are left justified. So, when you manually scan a log on a screen or on paper, you can just look down the left side of the page to quickly find all the messages.

A manual scan of the LOG in Exhibit 1 is easy. However, a manual scan of 250-page LOG (and they do exist) can take a lot of time and your manual scan can inadvertently miss a critical NOTE or WARNING pointing to a bug in your code.

A manual scan of the LOG given in Exhibit 1 might look something like what is displayed in Exhibit 2. Yet Exhibit 2 was created with a SAS program which found these messages. This program is presented in the next section and can save you time and headaches.

```
Exhibit 2: The Annotated LOG Outputted by subLOG0.sas (program in Exhibit 3)
-------------------------------------------------------------------------------
     FOLEY'S LIST OF CRITICAL LINES IN LOG
     ************************************
     (input file=C:\MYSAS\WORK\BAD_CODE.log)

NOTE: Variable U is uninitialized.
NOTE: Division by zero detected at line 5 column 9.
NOTE: Invalid second argument to function SUBSTR at line 8 column 8.
NOTE: Mathematical operations could not be performed at the following

  --------------END-OF-LOG---------------------
-------------------------------------------------------------------------------
```

PROGRAM TO SUBSET THE LOG

Exhibit 3 gives a SAS program that will take a file with a LOG extension (as in Exhibit 1) and create a corresponding file with a LG extension (see Exhibit 2) that contains only the "important" messages from the inputted LOG.

This program takes advantage of the fact that all the messages start in column 1 of the LOG. It is quite simple in construction. All the program subLOG0 does is to read each line from the LOG, and via one large IF statement at the end of the program, decide which lines to output. Which lines the program decides to output is discussed in the next section.

As you can see in Exhibits 2 and 3 that the program also adds three header lines and a footer line to the output file.

```
Exhibit 3: Program which Subsets the LOG in Exhibit 1 to become Exhibit 2
-------------------------------------------------------------------------------
* PROGRAM: subLOG0.sas;
* DESCRIP: Program to find the important messages in a SAS LOG file;
*          (User must change LET STEM= and RUN the program);
* AUTHOR: Malachy J. Foley;

%LET STEM=C:\MYSAS\WORK\BAD_CODE;
FILENAME IN_FILE "&STEM..log";
FILENAME OUT_FILE "&STEM..lg";

* Reading the .LOG file and outputting the .LG subfile;
DATA _NULL_;
  * In a LOG file the max LS=256, min LS=64;
  length text $256;
  infile IN_FILE lrecl=256 pad end=LAST;
  file   OUT_FILE;
  IF _N_=1 THEN DO;
```

```
   PUT "     FOLEY'S LIST OF CRITICAL LINES IN LOG";
   PUT "     ************************************";
   PUT "     (input file=&STEM..log)"/;
   END;

  input text $ 1-256;
  text=LEFT(text);

  * BIG IF which outputs only the important SAS messages;
  IF SUBSTR(text,1,4)  =  "*** "              OR
     SUBSTR(text,1,5)  =  "INFO:"             OR
     SUBSTR(text,1,6)  =  "ERROR:"            OR
     SUBSTR(text,1,8)  =  "WARNING:"          OR
     SUBSTR(text,1,13) IN("NOTE: Variabl",
                          "NOTE: Invalid",
                          "NOTE: Library",
                          "NOTE: MERGE s",
                          "NOTE: A hardw",
                          "NOTE: Charact",
                          "NOTE: Missing",
                          "NOTE: Divisio",
                          "NOTE: Mathema",
                          "NOTE: Interac"
                          )                  THEN PUT TEXT;

  IF LAST THEN PUT /"     --------------END-OF-LOG----------------------";
RUN;
-------------------------------------------------------------------------------
```

WHAT THE SUBSETTING PROGRAM LOOKS FOR IN THE LOG

When you manually view at a LOG, you are normally looking for messages that might indicate a problem with the corresponding program. What constitutes a problem varies from programmer to programmer and organization to organization. For example, some organizations allow some WARNING messages to be in the LOG's of their final products. Other organizations do not allow WARNING's in their production programs.

This paper suggests that all ERROR, WARNING and INFO messages that appear in a LOG are "important" and to be outputted to the annotated LOG file (the file with the LG extension). Furthermore, some NOTE's often indicate a critical condition in your program. The program subLOG0 (see Exhibit 3) looks for all ERROR, WARNING and INFO messages as well as the NOTE's given in Exhibit 4. (subLOG0 actually only looks for the first 13 characters of these NOTE's.) The letters XXX in Exhibit 4 represent the values that SAS places in the NOTE's to describe exactly what the NOTE is referring to.

Exhibit 4: List of Possible Critical NOTE's in a SAS LOG.

```
-------------------------------------------------------------------------------
 1) NOTE: Variable XXX is uninitialized.
 2) NOTE: Invalid XXX argument to function XXX at line XXX column XXX.
 3) NOTE: Library XXX does not exist.
 4) NOTE: MERGE statement has more than one data set with repeats of BY values.
 5) NOTE: A hardware font was specified that is not available on the device XXX.
 6) NOTE: Character values have been converted to numeric values at the places…
 7) NOTE: Missing values were generated as a result of performing an operation on
 8) NOTE: Division by zero detected at line XXX column XXX.
 9) NOTE: Mathematical operations could not be performed at the following places.
10) NOTE: Interactivity disabled with BY processing.
-------------------------------------------------------------------------------
```

IMPLEMENTING THE ANNOTATING PROGRAM

There are a number of ways to implement the subsetting program subLOG0 (Exhibit 3) in the SAS batch mode.

The simplest way is to change the %LET STEM= line in the program to give the path and file stem of the LOG you wish to subset, and then run subLOG0. You might want to use the program this way on your computer just to see how the program works. However, to use this implementation on a regular basis could become burdensome.

Another way to implement subLOG0 is to use the code within every program. This way the annotated LOG file is produced within the same SAS run as the LOG file. To apply this method, you must convert subLOG0 into a macro and include the macro in the original program. Then via PROC PRINTTO have the LOG messages redirected to some file other than the default LOG file. Close the non-default LOG with another PROC PRINTTO. Finally, call the subLOG0 macro to subset the non-default LOG. This method of using the annotating program allows you to create the LOG and LG files all in one step while executing the original program. But, it requires a little additional coding with each program.

Yet another way to apply subLOG0 is via your Operating System's macro facilities. This is the method preferred by the author and it is detailed in the next section.

AUTOMATING THE ENTIRE PROCESS

To get the annotated log in the batch environment, you need to do several things: run your original program, edit the subLOG0 program, run the subLOG0 program, and finally, use a text editor to display the annotated LOG file (LG). In many Operating Systems (OS) all of these functions can be automated via an OS macro facility. In Microsoft DOS (WINDOWS) all of the above steps can be placed in a DOS Batch Program and the entire process can be launched with a single DOS command like "SASF filename". Exhibit 5 presents such a Batch Program and the next section explains how it works.

(Note that the SASF Batch Program works in conjunction with a slightly different version of the subLOG0 program called subLOG (without the 0). subLOG is given in Exhibit 6.)

Exhibit 5: The SASf.BAT DOS Batch Program (works with subLOG.SAS in Exhibit 6)

```
@echo off
rem PROG = SASf.BAT
rem AUTHOR = Malachy J. Foley
rem DESCR  = A DOS Batch program to
rem             1) run a SAS program
rem             2) then run another SAS program called subLOG which
rem                eliminates extreneous infor from a sas log and
rem                leaves only messages that could be errors or problems
rem             3) then display the results of subLOG
rem
echo c:\mysas\work\%1 >c:\batfile.txt
c:
cd\program files\sas institute\sas\v8\
sas c:\mysas\work\%1
copy %1.* c:\mysas\work
del %1.*
sas c:\mysas\work\subLOG
del c:\bat\batfile.txt
cd\mysas\work
rem you can call your favorite text editor (including SAS) here
call notepad %1.lg
```

HOW THE SASF BATCH PROGRAM WORKS

This section provides a narrative of the steps the computer takes when the SASF program (Exhibit 5) is run. If you are familiar with DOS Batch Programs you may want to skip this section.

First SASF must be launched in a DOS WINDOW with a command "SASF BAD_CODE". The value that follows SASF is set to a DOS variable called %1. BAD_CODE.SAS (Exhibit 1) is the SAS program that will be processed by SASF. BAD_CODE.SAS is assumed to reside in a work directory/folder called C:\mysas\work. The user can change the work directory by changing SASF. The SASF.BAT file must be somewhere on the DOS PATH. DOS finds the BAT file and begins the execution.

The first @echo instruction in the BAT file (Exhibit 5) turns off printing to the DOS screen. If you want to debug the BAT file you might temporary remove this instruction. The next many rem instructions are simply comments or REMarks that the system ignores.

The next echo line says to write the string "c:\mysas\work\bad_code" to a file called c:\batfile.txt. This string will later be read by subLOG.sas (Exhibit 6).

The next two lines switch DOS to the directory/folder where the SAS executable resides. This folder is called c:\program files\sas institute\sas\v8. You will have to change these lines to indicate where SAS resides at your site.

The remaining lines are fairly self explanatory. They are regular DOS commands. Remember that %1 will be substituted with BAD_CODE in all cases (see above). The line "sas c:\mysas\work\%1" will execute SAS on file in the directory where SAS resides. The results (LOG file, etc.) are then copied to work directory. The results are then deleted from the SAS directory. subLOG.sas which resides in the work directory is then run. DOS is switched to the work directory. Finally, notepad (a text editor) is used to display the annotated LG file.

THE SUBLOG.SAS PROGRAM

The subLOG.sas program giving in Exhibit 6 is similar to the subLOG0.sas program (Exhibit 3) except at the beginning of the program the STEM macro variable value is read from the c:\batfile.txt file.

Exhibit 6: The subLOG.sas Program (works with the SASF.BAT in Exhibit 5)

```
* PROGRAM: subLOG.sas;
* DESCRIP: Program to find the important messages in a SAS LOG file;
*          (stand-alone version used by the SASf.bat DOS Batch program.);
* AUTHOR: Malachy J. Foley;

* Getting input file STEM from what the BATCH program wrote;
*   (Stem is the path and filename without the extension);
FILENAME IN_FILEs "C:\batfile.txt";
DATA _NULL_;
  length name $80;
  infile IN_FILEs lrecl=80 pad;;
  input name $ 1-80;
  call symput('STEM',TRIM(name));
RUN;

FILENAME IN_FILE "&STEM..log";
FILENAME OUT_FILE "&STEM..lg";

* Reading the .LOG file and outputting the .LG subfile;
DATA _NULL_;
```

```
  * In a LOG file the max LS=256, min LS=64;
  length text $256;
  infile IN_FILE lrecl=256 pad end=LAST;
  file   OUT_FILE;
  IF _N_=1 THEN DO;
     PUT "     FOLEY'S LIST OF CRITICAL LINES IN LOG";
     PUT "     ************************************";
     PUT "     (input file=&STEM..log)"/;
     END;

  input text $ 1-256;
  text=LEFT(text);

  * BIG IF which outputs only the important SAS messages;
  IF SUBSTR(text,1,4)  =  "*** "              OR
     SUBSTR(text,1,5)  =  "INFO:"             OR
     SUBSTR(text,1,6)  =  "ERROR:"            OR
     SUBSTR(text,1,8)  =  "WARNING:"          OR
     SUBSTR(text,1,13) IN("NOTE: Variabl",
                          "NOTE: Invalid",
                          "NOTE: Library",
                          "NOTE: MERGE s",
                          "NOTE: A hardw",
                          "NOTE: Charact",
                          "NOTE: Missing",
                          "NOTE: Divisio",
                          "NOTE: Mathema",
                          "NOTE: Interac"
                          )                  THEN PUT TEXT;

  IF LAST THEN PUT /"     --------------END-OF-LOG----------------------";
RUN;
```

FINAL NOTES ON IMPLEMENTATION

If you are planning on implementing the programs given Exhibits 5 and 6, it might be helpful to have a list of where each file of the system is located in the directories/folders. Exhibit 7 is such a list.

Exhibit 7: List of File Folders Where Different Files Should Reside

FILE	DIRECTORY/FOLDER
SASF.BAT	Anywhere on the DOS PATH
BAD_CODE.SAS (target file)	Work Directory
subLOG.SAS	Work Directory
All results (LST LOG LG)	WORK Directory
NOTEPAD.exe	Anywhere on the DOS Path

CONCLUSION

Using the SAS program given in Exhibits 6 along with the DOS Batch program given in Exhibit 5 can greatly simplify the implementation of batch SAS. With one simple batch command, the two programs run your SAS program, subset the LOG to critical messages and display the subsetted LOG for your review. Moreover, these programs will save the programmer time and guarantee that all important LOG messages are found no matter how large the LOG.

These two programs can be modified in many ways to the user's specification. For example, the definition of what are important messages in a SAS LOG can be defined by the user in the IF statement of

subLOG.sas. Also, the user can adapt the programs to other operating systems or environments.

All in all, the programs given Exhibits 5 and 6 can cut any SAS LOG down to size.

TRADEMARK INFORMATION

ABOUT THE AUTHOR

Mal Foley started programming computers in high school and never stopped. His career includes being an international computing consultant, a university professor, and the CEO of his own computing company. Currently, Mal is a senior SAS programmer/analyst in the Department of Biostatistics at the University of North Carolina at Chapel Hill. He actively participates in local, regional, and national SAS user's groups. He is also a part-time SAS trainer and gives in-house SAS seminars around the county.

AUTHOR CONTACT

The author welcomes comments, questions, corrections and suggestions. You may contact the author at:

Malachy J. Foley
2502 Foxwood Dr.
Chapel Hill, NC 27514

Email: FOLEY@unc.edu