Paper CC-037

SAS® Log Summarizer – Finding What's Most Important in the SAS® Log

Milorad Stojanovic
RTI International
Education Surveys Division
RTP, North Carolina

ABSTRACT

Validity of SAS® programs is an important task. Lots of information about the execution of SAS programs can be found in SAS® Log file. Carefully examination of the SAS log is a mandatory task, but it can be cumbersome and tedious for very large logs. Log Summarizer is a tool designed to reduce the burden of this review. It can be used to extract the most important information (ERRORs, WARNINGs, and specific NOTEs), while preserving section indicators, statement sequence, original line numbers and message text. Log Summarizer's output is in RTF format and will provide programmer and/or QA analyst with a condensed version of the information they are most interested in reviewing.

KEYWORDS: SAS LOG, ERROR, WARNING, NOTE

INTRODUCTION

To do programming jobs correctly we should check the results of our programs. Author points to the papers by Rick Mitchell (1) and Howard and Gayari (2) as ones everybody should take time to read. Both papers give excellent guidelines how to validate your results on the more broader basis than only checking SAS LOG. As users of SAS, analyst, programmers, statisticians and others are keenly interested in the successful operation of SAS programs, and are generally satisfied to have their programs execute successfully (resulting in new files, reports or both) while paying no attention to the SAS LOG. Other users, however, will check the program LOG for ERRORs only. And still others will check all messages in the SAS LOG. The most thorough and complete check would include checking of results of each step, but it is often a cumbersome and lengthy process. For example, newly created data sets which end up with 0 observations and N variables could indicate possible errors in data processing , which could be the first sign that something went wrong.

In the papers already presented at various SAS User Group (SUG) meeting (see references) authors used a more or less generalized approach to analyzing SAS LOGs and to extract important information from their perspective. Mainly they were dealing with ERROR, WARNING and selected NOTE messages. This paper takes the review a step further, by recommending additional checking for missing BY statements, checking for data set(s) with 0 observations, and checking for messages that another process was locking the data set.

DESCRIPTION OF PROGRAM

The goal was to prepare a condensed report structured in several sections with the hierarchy of messages from the most severe to the simply informative. SAS LOGs were treated as any other text file, with a log file line size of 200 characters. The SAS logs were searched for three keywords: ERROR, WARNING, and NOTE. NOTE messages were differentiated, as some of them are purely informational with no significant value for finding mistakes in program, whereas others could often (but not always) point to some incorrect behavior of the program during the execution phase. To make intervention in the program code as small as possible, a "%INCLUDE" statement was used to isolate specific NOTE messages. In this way, maintenance of the program was much easier. Users are advised to add or remove NOTEs messages and to choose an appropriate location for report as well as for the input SAS LOG.

It is important to note that it could be a highly dangerous mistake to exclude a "BY" statement in MERGE. Although the program will seemingly execute successfully, the results could include some 'bad' merging of input data sets. Moreover SAS would not give any ERROR or WARNING message (if you don't have any other mistakes in said code), but would only yield three naïve NOTE messages such as the following:

“NOTE: There were 3 observations read from the data set WORK.ONE.
NOTE: There were 3 observations read from the data set WORK.TWO.
NOTE: The data set WORK.THREE has 3 observations and 3 variables.“

These seemingly innocuous notes hide a potential disaster. The output will include a new data set in the output, but merging is now highly problematic. **The most important thing to note is that this is not VISIBLE at all**. Consider the following:

Data set ONE

Obs	a	b
1	1	15
2	2	25
3	4	45

Data set TWO

Obs	a	c
1	1	115
2	2	225
3	3	335

No BY statement

Obs	a	b	c
1	1	15	115
2	2	25	225
3	3	45	335

With BY statement

Obs	a	b	c
1	1	15	115
2	2	25	225
3	3	.	335
4	4	45	.

PROGRAM CODE

```
%let saslog_location = C:\SESUG08\TESTLOG.log;
%let saslog_notes    = C:\SESUG08\NOTE_messages.txt ;
%let report_location = C:\SESUG08 ;

%macro SASLOG_Summarizer(saslog_location=, saslog_notes=, report_location=) ;
filename infl "&saslog_location." ;
options ls=80 ps=50 nodate nonumber ;

proc format ;
value Typefm  1 = 'ERROR'
              2 = 'WARNING'
              3 = 'Spec. NOTEs'
              4 = 'NOTE' ;
run ;

data mistakes(keep=type txt line_counter) ;
retain line_Counter 0 ;
length txt $ 190 ;
infile infl lrecl=200 pad ;
input @1 intext $200. ;
line_counter + 1 ;
if index(intext, 'ERROR:')>0 then do ;
   Type = 1 ;
   pos = index(intext, '2E0D'X) ;
   txt = substr(intext, 8, pos - 8) ;
   output ;
end ;
else if index(intext, 'WARNING:')>0 then do ;
   Type = 2 ;
   txt = substr(intext, 10) ;
   output ;
end ;
else if index(intext, 'NOTE:')>0 then do ;
     if index(intext, 'The data set')>0 and
     index(intext, 'has 0 observations')>0 then do ;
        Type = 3 ;
        txt = substr(intext, 6) ;
        output ;
    end ;
end ;
if index(intext, 'NOTE:')>0 then do ;
    %include "&saslog_notes." ;
```

```
    Type = 4 ;
    txt = substr(intext, 7) ;
    output ;
    end ;
end ;
run ;

* Additional step for checking of missing BY statements in MERGE. ;
data mistakes1(keep=type txt line1) ;
retain line_Counter Merge_flag Data_flag Proc_flag RUN_flag
                Merge_loc  Data_loc  Proc_loc  Run_loc  BY_flag 0;
length txt $ 190 ;
infile infl lrecl=200 pad ;
input @1 intext $200. ;
intext = upcase(intext) ;
line_counter + 1 ;
if index(intext, 'NOTE:')=0 and
   index(intext, 'ERROR:')=0 and
   index(intext, 'WARNING:')=0 then do ;
   if index(intext, ' MERGE ')>0 and index(intext, '*')=0 then do ;
      Type = 3 ;
      merge_loc= line_counter ;
      merge_flag = 1 ;
      Data_loc=0;  Proc_loc=0;  Run_loc=0 ; Proc_flag = 0 ;
      data_flag = 0 ; run_flag = 0 ; by_flag = 0 ;
   end ;
   if merge_flag = 1 then do ;
      if index(intext, ' BY ')>0 then BY_flag = 1 ;
      if index(intext, ' PROC ')>0 then do ;
          Proc_flag = 1 ;
          Proc_loc  = line_counter ;
      end ;
      if index(intext, ' DATA ')>0 then do ;
         data_flag = 1 ;
         Data_loc  = line_counter ;
      end ;
      if index(intext, ' RUN ;')>0 then do ;
         run_flag = 1 ;
         Run_loc  = line_counter ;
      end ;
        if BY_flag=0 and
          (Data_flag=1 or Run_Flag=1 or Proc_Flag=1) and
           (0<=(Line_counter - merge_loc)<=6) then do ;
          type = 3 ;
          txt = 'No BY statement for preceding MERGE statement' ;
          line1 = MERGE_loc ;
          output ;
          merge_flag=0 ; Proc_flag = 0 ; data_flag = 0 ; run_flag = 0 ; by_flag = 0 ;
          data_loc = 0 ; proc_loc = 0 ; merge_loc = 0 ; run_loc = 0 ;
        end ;
   end ;
end ;
run ;

proc sort data=mistakes ;
   by type line_counter ;
run ;

data mistakes ;
set mistakes mistakes1(rename=(line1=line_counter)) ;
by type line_counter ;
run ;
```

```
ods rtf file="&Report_location\SASLOG_Summarizer.rtf" style=banker ;
proc report data=mistakes nowd split='~' ;
title  J=C "Report from SASLOG Summarizer for:" ;
title2 J=C "&saslog_location.";
column type line_counter txt ;
define type         / group order=internal f=typefm. 'Type' ;
define line_counter / display f=6.            'Line # in~SAS LOG' ;
define txt          / display width=80 /*f=$90.*/ flow  'Original SAS log message' ;
run ;
ods rtf close ;

%mend SASLOG_Summarizer;

%SASLOG_Summarizer(saslog_location=&saslog_location.,
                   saslog_notes=&saslog_notes., report_location=&report_location.)
```

Content of **include** file could be same like in this file or with different content.

```
if index(intext, 'Invalid') > 0 or
   index(intext, 'W.D format') > 0 or
   index(intext, 'is uninitialized') > 0 or
   index(intext, 'repeats of BY values') > 0 or
   index(intext, 'Mathematical operations could not') > 0 or
   index(intext, 'Missing values were') > 0 or
   index(intext, 'Division by zero') > 0 or
   index(intext, 'MERGE statement') > 0 or
   index(intext, 'Character values have') > 0 or
   index(intext, 'values have been converted') > 0 or
   index(intext, 'Interactivity disabled with') > 0 or
   index(intext, 'No observation') > 0
   then do ;
```

Program execution with input of over 17K+ lines (TESTLOG.log) was below 1 (one) second on author's PC. With separation of LOG keyword messages into separate file (NOTE_Messages.txt) program / macro gets enough of flexibility and could be tailored to the need of each user.

EXAMPLE OF REPORT

Report from SASLOG Summarizer for:
C:\SESUG08\TESTLOG.log

Type	Line # in SAS LOG	Original SAS log message
ERROR	17369	File NEW.TEST1.DATA does not exist
	17372	No data set open to look up variables
	17404	Errors printed on pages 304,305
WARNING	2368	Multiple lengths were specified for the BY variable PR by input data sets. This may cause unexpected results.
	16277	The variable RACE in the DROP, KEEP, or RENAME list has never been referenced.
	16278	The variable SEX in the DROP, KEEP, or RENAME list has never been referenced.
	17374	No data sets qualify for WHERE processing.

Type	Line # in SAS LOG	Original SAS log message
Spec. NOTEs	2921	No BY statement for preceding MERGE statement
	3789	The data set WORK.TEST2 has 0 observations and 5 variables.
	4943	No BY statement for preceding MERGE statement
NOTE	929	Invalid argument to function INPUT at line 3263 column 25.
	988	Missing values were generated as a result of performing an operation on missing values.
	991	Mathematical operations could not be performed at the following places. The results of the operations have been set to
	1517	Numeric values have been converted to character values at the places given by: (Line):(Column).
	1519	Character values have been converted to numeric values at the places given by: (Line):(Column).
	4280	No observations in data set WORK.TEST3.

CONCLUSION

With investing of relatively moderate efforts it was possible to get useful reports which directs SAS users to the most important messages in the SAS LOG and provided an additional tool for QA of SAS programs. Absolute position of message in SAS LOG gives quick access to the point of possible mistake.

ACKNOWLEDGEMENTS

Author would like to thank James Isaacs and Peter Einaudi from Education Studies Division (RTI) for all their help, comments, and support in producing this paper.

REFERENCES

1. Mitchell, Rick M., Finding Your Mistakes Before They Find You: A Quality Approach For SAS® Programmers http://www.nesug.info/Proceedings/nesug06/as/as04.pdf , NESUG 2006

2. Howard Neil, Gayari Michelle, Validation, SAS, and the Systems Development Life Cycle: An Oxymoron? http://www.lexjansen.com/pharmasug/2000/dmandvis/dm09.pdf , Pharmasug 2000

3. Truong Sy, Making Code Review Painless, http://www.meta-x.com/wuss12_making_code_review_pain.pdf , WUSS 12

4. Smoak Carey G, A Utility Program for Checking SAS Log Files, http://www2.sas.com/proceedings/sugi27/p096-27.pdf SUGI 27

5. Gregg Keith M. Gershteyn Yefim, Checking and Tracking SASâ Programs Using SASâ Software, http://www.units.muohio.edu/doc/sassystem/sugi25/24/AppDevel/p28-24.pdf SUGI 25

6. Augustine Aaron, Dutta Prasenjit, You've Got E-Mail: Automatic Log Checking Via E-mail Notification http://www2.sas.com/proceedings/sugi31/128-31.pdf SUGI 31

7. Li Tianshu, A Macro to Report Problematic SAS Log Messages in a Production Environment http://www.nesug.org/proceedings/nesug01/cc/cc4008.pdf NESUG 2001

8. Markovitz Heidi, SAS Completion Codes to Make Complex Programs Run Smoothly http://analytics.ncsu.edu/sesug/2002/CC07.pdf SESUG 2002

9. Humphreys Suzanne, %LOGCHECK: a Convenient Tool for Checking Multiple Log Files http://www.lexjansen.com/pharmasug/2008/cc/cc02.pdf Pharmasug 2008

10. Foley Malachy J., Cutting the SAS® LOG down to size,
http://analytics.ncsu.edu/sesug/2004/SY05- Foley.pdf SESUG 2004

11. Mason Philip, SASTip77 - Automatic checking of the LOG,
http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0310d&L=sas-l&P=16971

12. Mark Terjeson,Log file reading program,
http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0411E&L=sas-l&P=16645

13. Fehd Ronald, Tip: macro LOGSAVE v1,
http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0110d&L=sas-l&F=&S=&P=8887

14. Sherman Paul, Intelligent SAS Log Manager,
http://analytics.ncsu.edu/sesug/2007/AD15.pdf SESUG 2007

DISCLAIMER

All opinions and suggestions stated in the paper on how to check and validate SAS LOGs do not necessarily reflect the opinions and suggestions of RTI International. Use any of commercial products mentioned in references for checking SAS LOG is the responsibility of the individual user.

CONTACT INFORMATION

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

Milorad Stojanovic
Education Surveys Division
RTI International
3040 Cornwallis Rd
RTP, NC, 27909
Work Phone : (919) 541-7376
E-mail: milorad@rti.org

TRADEMARK INFORMATION