Paper 168

Quote the SASLOG®

Andrew T. Kuligowski, Independent Consultant

ABSTRACT

"For every action, there is an equal and opposite reaction." Sir Isaac Newton was talking about physics, but the concept also applies to other areas of life - including quotation marks in SAS® code. Unfortunately, SAS coders do not always properly provide balanced quotation marks while coding. SAS will detect possible occurrences of this problem, signaling its concern with a SASLOG message:

```
WARNING:  The quoted string currently being processed has become more than
          262 characters long.
          You might have unbalanced quotation marks.
```

This presentation contains a few coding suggestions to identify unbalanced quotation marks in SAS code.

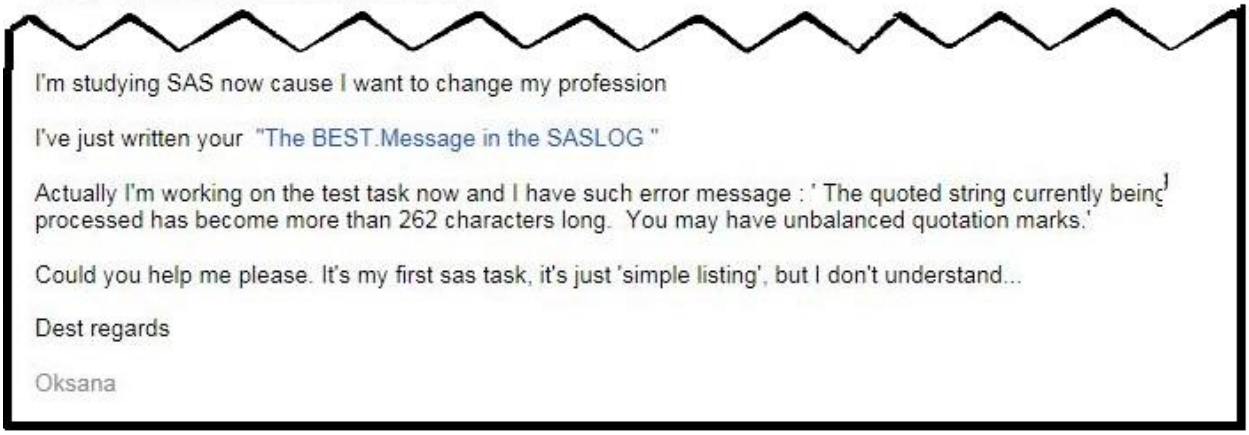
KEYWORDS

SASLOG, Base SAS, SASLOG, Quotation Marks, “quoted string”

INTRODUCTION / BACKGROUND

Breaking through the proverbial fourth wall … I’m sure that most of the people reading this paper have received an email that begins similar to the following:

Unlike most other such emails, this one made it through my SPAM filter. As such, I got curious and decided to launch and read it. The next line introduced a plot twist not typically found in such emails:



This is NOT how these types of emails typically go at least not in my experience! The question got me curious, so I began to look into the problem: How can you quickly identify the cause of an unbalanced quotation mark?

STEP 0: FIX THE IMMEDIATE PROBLEM

A doctor once remarked to me that, while he'd like to identify the cause of a particular patient's challenging malady, he'd prefer NOT to do it at an autopsy. Therefore, he had to deal with reversing the immediate problematic symptoms before he could deal with the long-term issue of identifying the actual cause and suggesting a cure. The same, to a much less significant extent, is true for the unbalanced quote.

Most SAS users have code to close an unidentified open quote, comment, or loop. The following is the one that I use:

```
;  *';  *";  */;
ODS _ALL_ CLOSE;
quit;  run;  %MEND;
data _NULL_;  putlog "DONE";  run;
```

It may be necessary to execute this code multiple times – for example, if you accidentally type `%MACRO` when you intended to type `%MEND` to terminate a macro definition, you will have TWO open macro definitions. It will be necessary to execute the code snippet twice – or possibly more. The DATA _NULL_ at the end provides visual evidence that you've resolved all of the open statements / constructs.

APPROACH 1: BRUTE FORCE: COUNT THE QUOTATION MARKS IN THE SASLOG

I call this the "Brute Force" method because it is based on the simple assumption that you expect to find quotation marks in an even number.

First, determine the line in which the first "unbalanced quotes" warning occurs in the SASLOG. We will do this by reading our SASLOG itself into SAS, and looking for a string from the warning message:

```
DATA TEMP ;
   INFILE  "/SASLOGs/SAMPLE-OpenEndedQuote.log";
   INPUT ;
   IF  FIND( LOWCASE( _INFILE_ ), '262 characters' )  THEN LineNum_Error = 1 ;
   CALL  SYMPUT( "ERR_LINENO", _N_ );
RUN;
```

No matter how the long the SASLOG happens to be, we know that the situation that triggered the message has to occur prior to this message. (The OBS= option below is not required but may help speed the process along if the SASLOG is lengthy – this will stop the DATA step once the offending line is found, assuming that the previous DATA step has been invoked. The DATA step could also be halted without actually executing the prior step by putting a STOP statement in the "262 characters" IF statement in THIS DATA step.)

```
DATA TEMP ;
   LENGTH Line_of_Data $ 200. ;
   INFILE "/SASLOGs/SAMPLE-OpenEndedQuote.log" OBS=&ERR_LINENO.;
   INPUT ;
   IF  FIND( LOWCASE( _INFILE_ ), '262 characters' )  THEN  LineNum_Error = 1 ;
   Line_of_Data = _INFILE_;
   Find_SingleQuote = FIND( _INFILE_, "'" );
   Find_DoubleQuote = FIND( _INFILE_, '"' );
   IF   LineNum_Error   OR   Find_SingleQuote  OR  Find_DoubleQuote  THEN
      OUTPUT;
RUN;
```

We *may* have enough information at this point to solve our problem. By browsing the TEMP dataset in SAS, we can make a few helpful assumptions:

1) As stated earlier the Warning message in the SASLOG is the point from which we want to work BACKWARDS. Ignore all the lines after the message is identified.
2) We can manually count the number of quotation marks in each line.
3) We expect an even number of quotation marks – single and double being treated separately, of course. Any individual line could have an odd number of quotation marks, but the odd numbers in one line must be offset by an odd number in another.

	Line_of_Data	LineNum_Error	Find_SingleQuote	Find_DoubleQuote	Log_LineNum
1	initialization. Edit the file "news" in the "misc/base" directory to		0	32	23
2	The command line option "-nonews" will prevent this display.		0	25	25
3	2 RETAIN Char_Constant "X ;		0	30	73
4	15 PUTLOG @ 1 'Number:' @ 9 _N_ 2. @ 12 Order_Random 2.		21	0	86
5	WARNING: The quoted string currently being processed has become more than 262 characters long. You might have unbalanced quotation marks.	1	0	0	87
6	16 @ 15 '(' @ 16 Sort_Order 11.8 @ 25 ')' ;		21	0	88
7	21 RETAIN Char_Constant 'X';		30	0	93
8	34 PUTLOG @ 1 'Number:' @ 9 _N_ 2. @ 12 Order_Random 2.		21	0	106
9	35 @ 15 '(' @ 16 Sort_Order 11.8 @ 25 ')' ;		21	0	107
10	39 RETAIN Char_Constant 'X';		30	0	111
11	52 PUTLOG @ 1 'Number:' @ 9 _N_ 2. @ 12 Order_Random 2.		21	0	124
12	53 @ 15 '(' @ 16 Sort_Order 11.8 @ 25 ')' ;		21	0	125
13	55 "; run;		0	6	127

1) Find the message in the SASLOG

2) Ignore all of the lines after the offending message is located.

3) Manually count the quotation marks.

	Line_of_Data	LineNum_Error	Find_SingleQuote	Find_DoubleQuote	Log_LineNum
1	initialization. Edit the file "news" in the "misc/base" directory to		0 0	4 32	ok 23
2	The command line option "-nonews" will prevent this display.		0 0	2 25	ok 25
3	2 RETAIN Char_Constant "X ;		0 0	1 30	« aha! « 73
4	15 PUTLOG @ 1 'Number:' @ 9 _N_ 2. @ 12 Order_Random 2.		2 21	0 0	ok 86
5	WARNING: The quoted string currently being processed has become more than 262 characters long. You might have unbalanced quotation marks.	1	0	0	87
6	16 @ 15 '(' @ 16		21	0	88

4) Note any lines with odd numbers of quotation marks that are not offset by subsequent numbers of quotation marks.

APPROACH 2: LET SAS COUNT THE QUOTATION MARKS IN THE SASLOG

The first approach can be executed fairly quickly, but it is too manually dependent. It identifies the lines containing the single quotes and double quotes, but it then requires a human being to manually count those quotation marks. This may take some time if the log is well populated with quotation marks, and it lays open the significant risk of human error should the person mis-count.

To combat this weakness, we will write a routine that performs the counts for us. It will identify the lines containing either single or double quotes (or both) and will keep count of each. Should the count be an odd number, it will report the line to the analyst for further examination.

```
DATA TEMP2;
   INFILE "/SASLOGs/SAMPLE-OpenEndedQuote.log";
   INPUT ;
   Line_of_Data = _INFILE_;
   Log_LineNum  = _N_ ;
   Find_SQuote = 0 ;   Count_SQuote = 0 ;
   Find_DQuote = 0 ;   Count_DQuote = 0 ;

   DO UNTIL( NOT( Find_SQuote ) );
      Find_SQuote = FIND( Line_of_Data, "'", Find_SQuote+1 );
      IF  Find_SQuote THEN Count_SQuote = Count_SQuote + 1;
   END;
   DO UNTIL( NOT( Find_DQuote ) );
      Find_DQuote = FIND( Line_of_Data, '"', Find_DQuote+1 );
      IF  Find_DQuote THEN Count_DQuote = Count_DQuote + 1;
   END;


   Unbalanced_SingleQuote_IND = MOD( Count_SQuote, 2 );
   Unbalanced_DoubleQuote_IND = MOD( Count_DQuote, 2 );



   IF  Count_SQuote > 0  OR  Count_DQuote > 0  THEN  DO;
      PUTLOG  @  1 Log_LineNum    5.  @  6 ":"
              @  8 Count_SQuote=  2.
              @ 30 Count_DQuote=  2.  @ ;
      IF  Unbalanced_SingleQuote_IND  THEN
         PUTLOG @ 52 "Check Single Quote"  @ ;
      IF  Unbalanced_DoubleQuote_IND  THEN
         PUTLOG @ 72 "Check Double Quote"  @ ;
      PUTLOG /  @ 11  "SOURCE: " Line_of_Data ;
   END;
RUN;
```

1) Initialize the Yes/No flags AND the quote mark counters.

2) Count the single quotes on each line.

3) Count the double quotes on each line.

4) Determine if there are potentially unbalanced quotes by determining if there is an even (expected) or odd (not) number of each.

5) Report on the lines containing quotes. Call out any that have unbalanced quotes for potential further examination.

```
23: Count_SingleQuote=0  Count_DoubleQuote=4
25: Count_SingleQuote=0  Count_DoubleQuote=2
73: Count_SingleQuote=0  Count_DoubleQuote=1     Check Double Quote
    2      RETAIN Char_Constant "X ;
86: Count_SingleQuote=2  Count_DoubleQuote=0
87: Count_SingleQuote=0  Count_DoubleQuote=0
```

The example above identifies the same potentially offending line that was flagged by our part-SAS, part-human approach earlier. However, the SAS routine does the work and only involves the human observer once it is done identifying lines with quotation marks, counting them, and subsequently flagging potentially suspicious lines of code.

APPROACH 2A: LET SAS COUNT THE QUOTATION MARKS IN THE ORIGINAL CODE

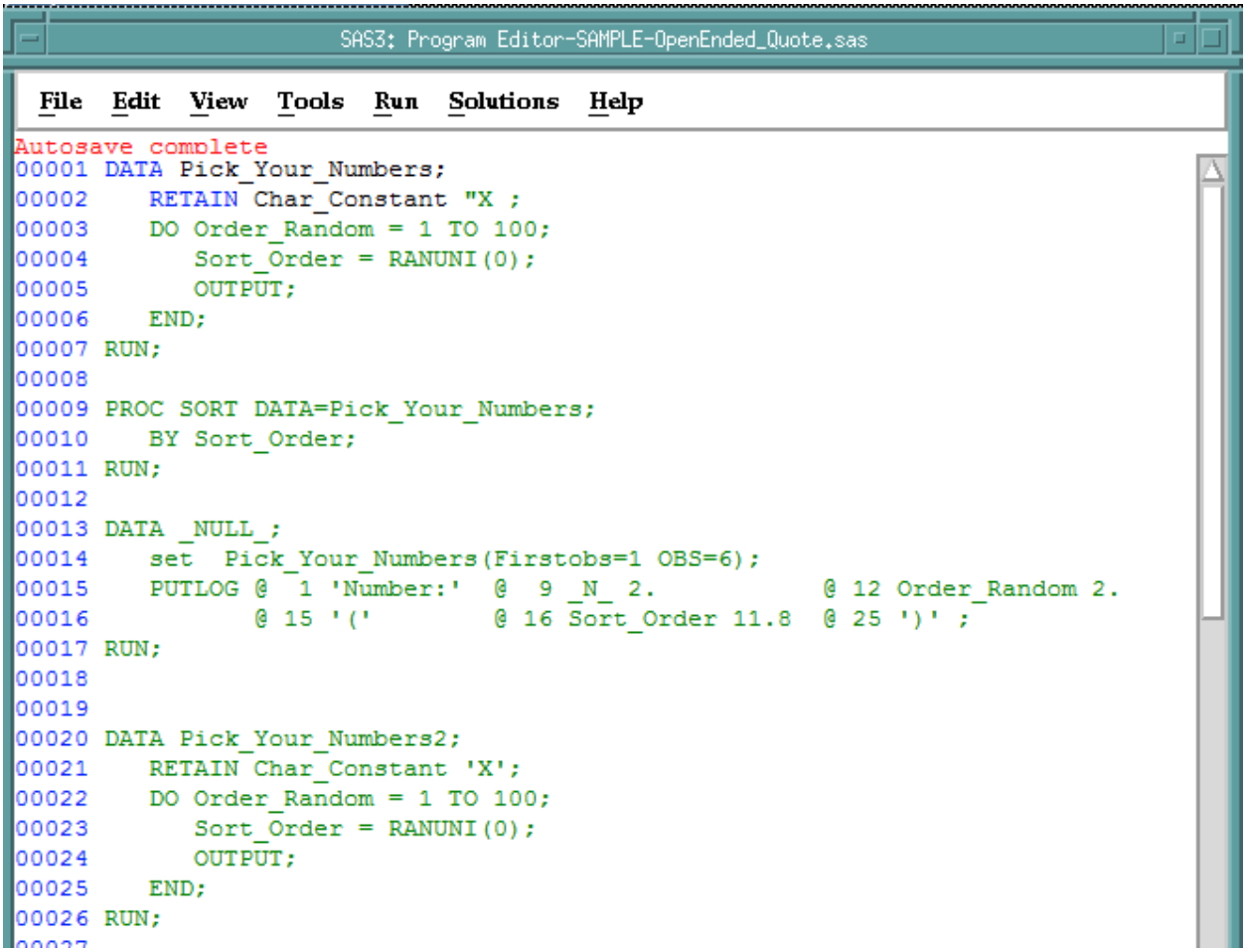
Approach 2A is essentially the same as Approach 2 – let a SAS routine read some text and call out potentially unmatched quotation marks. HOWEVER, this approach reads in the original routine itself, rather than the SASLOG. The benefit to this approach is that you do not have to execute the routine prior to examining it. The drawbacks are:

a) The analyst typically doesn't realize they have unbalanced quotation marks until they've executed their routine, and

b) IF the code is enhanced to stop at the point where the "unbalanced quotation marks" WARNING is found to save processing time, this approach will not work. That message only appears in the SASLOG.

STEP 3: LET SAS DO *ALL* THE WORK FOR YOU

Most of this work can be rendered as a classroom exercise if the reader is using SAS' Enhanced Editor. The Editor identifies comments – and quotations – by turning the contents green. An extended quantity of green lines most likely means that a quotation mark has been omitted:



It sounds obvious, but DO NOT execute code with an excessive number of green lines ~~and~~ unless you are certain that it does not contain unbalanced quotes or open comments.

CONCLUSION

This is a quick look at writing and employing SAS ad hoc routines to examine SAS code and SASLOGs for unbalanced quotes. The approach is basic and is not meant to describe fully functional and fully debugged utilities. (Indeed, a line such as *'; or *"; mentioned early in the presentation, would be called out as a potential error by the routines in this presentation.) The goal is to simply address an issue – in these cases, it is not necessary to "over-code" for potential complications that do not exist in the problem being examined. The approach of examining SAS code and SASLOGs can easily be extended to other types of errors.

It is hoped that the reader will take the spirit of the presentation – whether they use the actual code presented or not – and use it to help solve their own programming issues in their own situation.

ACKNOWLEDGMENTS

A special “Thanks” goes out to Oksana Frolova for triggering this investigation, and for her continued friendship since this initial inquiry. Also, a tip of the hat to the Toronto Area SAS Society (TASS), whose leadership first agreed that this particular problem might be worth exploring and documenting in a more conventional format.

CONTACT INFORMATION

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

Andrew Kuligowski
E-mail: KuligowskiConference@gmail.com