Paper 068-2011

You've Got ERROR

Ronald R. Palanca, Mathematica Policy Research, Princeton, NJ

Abstract

This paper presents a convenient tool to help SAS® programmers debug batch submitted SAS programs. After the execution of the program, a pop-up screen alerts the SAS programmer to any errors, warnings, uninitialized variables, or unwanted messages in the log. This program automatically gives SAS programmers a heads up about unwanted messages and specifically identifies the problems.

Introduction

Going through long program logs and searching for errors, warnings, uninitialized variables, and other relevant messages sometimes can be mind-numbing. Having a tool that will automatically tell you what the unwanted messages are in your SAS log would really come in very handy.

If included at the end of the SAS program, this MPR_Checklog macro searches for predefined messages in the SAS log. The output of the macro is hard to miss because a screen pops if the macro finds something.

Programmers can write separate SAS programs to scan through their SAS logs for errors. This macro incorporates that process so programmers receive an immediate alert after their program runs in batch mode.

How does it work?

The macro incorporates predefined words or phrases and uses them as the keywords to be scanned for in the SAS log. The macro can be customized to search for specific words or phrases in the SAS log that is automatically created when submitting a SAS program in batch mode.

The challenge here is that the SAS log is not created until after the execution of the SAS program. Because this macro is executed within the program, it is impossible to search for keywords if there is no output log yet. To get around this, the macro uses the Windows X command just before the execution of the other statements in the macro. The macro makes a copy of the preliminary log and saves it as a text file. This text file contains the SAS log up to the point at which the macro was executed.

```
X "COPY &INDAT..LOG &INDAT.2.LOG";
```

Some programs might have large logs so the sleep command pauses the execution of the macro to complete copying the preliminary log to a text file. This process can also be customized.

```
DATA _NULL_;
X=SLEEP(3);
```

The next process creates a temporary SAS dataset from the text file containing the log. Each row in the text file is an observation of a variable called Rows. This dataset will be filtered to contain only the observations that have the predefined words or phrases of interest. Here is an example of the code and the predefined words or phrases:

```
DATA CHECKLOG;
  ..............
  FLGERROR = 0;
 IF SUBSTR(ROWS,1,5)='ERROR' OR SUBSTR(ROWS,1,7)='WARNING'
 OR INDEX(UPCASE(ROWS),"UNINITIALIZED") > 0
 OR INDEX(UPCASE(ROWS),"_ERROR_") > 0
 OR INDEX(UPCASE(ROWS),"REPEATS OF BY VALUES") > 0
 OR INDEX(UPCASE(ROWS),"EXTRANEOUS") > 0
 OR INDEX(UPCASE(ROWS),"INVALID DATA FOR") > 0
 OR INDEX(UPCASE(ROWS),"SAS SYSTEM STOPPED PROCESSING") > 0
 OR INDEX(UPCASE(ROWS),"INVALID ARGUMENT") > 0
 OR INDEX(UPCASE(ROWS),"ODS PDF PRINTED NO OUTPUT") THEN OUTPUT;
```

The data in the checklog dataset might have quotation marks that affect the output of the macro, so we convert them to blanks. This can also be customized.

```
ROWS = TRANSLATE(ROWS,' ',"", " ","'");
```

The next step is to delete duplicate rows and to assign a number to each row of data. This variable will be called “count.” We also check if the checklog dataset has observations or not.

```
PROC SORT NODUPKEY; BY ROWS;

DATA CHECKLOG;
  SET CHECKLOG;
  COUNT + 0;
  BY ROWS;
  IF FIRST.ROWS THEN COUNT = COUNT + 1;

PROC SQL NOPRINT;
   SELECT COUNT(*) INTO: NUMOBS
   FROM CHECKLOG
   QUIT; ;
```

If the checklog dataset has no observations, we exit the macro. Otherwise, using the SAS CALL SYMPUT routine, macro variables are assigned to each value of the variables Rows and Count.

```
%IF &NUMOBS GE 1 %THEN %DO;

    DATA _NULL_;
      SET CHECKLOG END=LAST;
      IF LAST THEN DO;
         CALL SYMPUT('CNT', PUT(COUNT + 5, 3.));
      END;

    DATA _NULL_;
      SET CHECKLOG ;
      CALL SYMPUT('CNTERR'||LEFT(_N_ + 5), ROWS);
```

The data is output from the checklog dataset using %WINDOW from the SAS macro facility. Because of a limit to the number of rows that can be output from this macro facility, the number of messages that can be shown on the alert is limited to five.

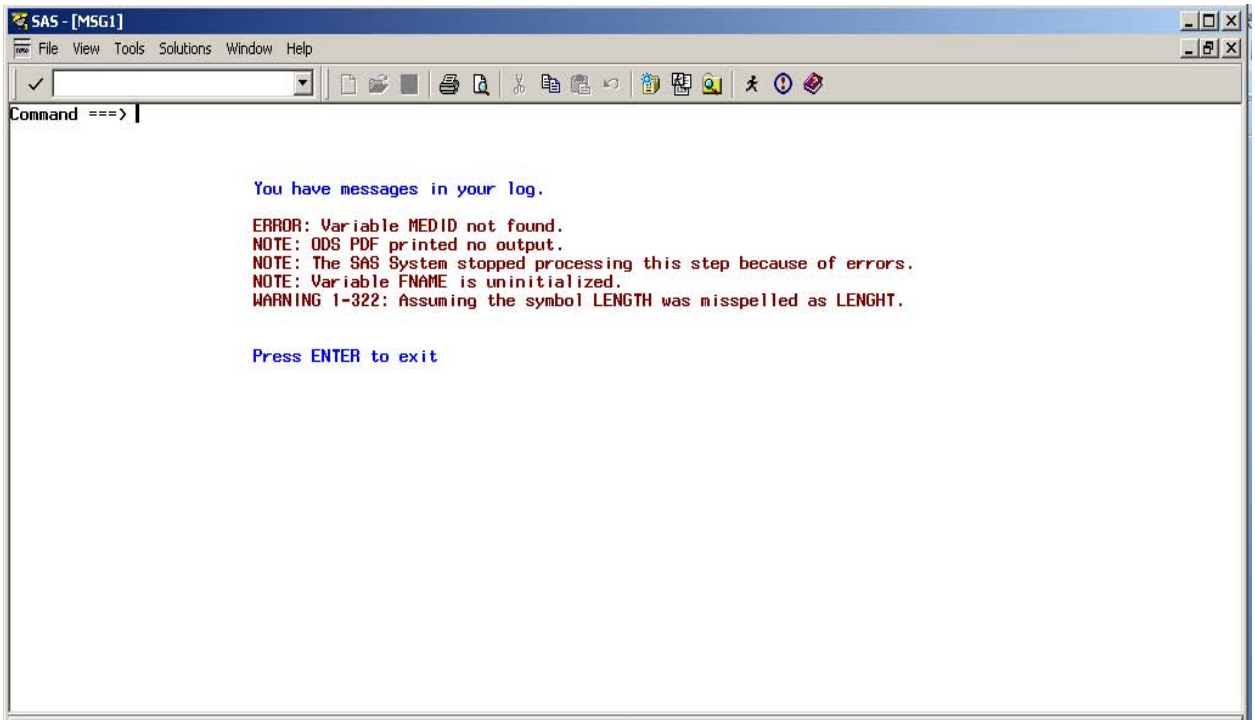
```
%MACRO RPT();
    %IF &NUMOBS. GE 5 %THEN %DO;
      %DO _I_ = 6 %TO 10 ;
          #4  @26 "You have messages in your log." Color  = BLUE
          #&_I_ @26 "&&CNTERR&_I_." color = RED
      %END;
   %END;
   %ELSE %IF &NUMOBS. < 5 %THEN %DO;
      %DO _I_ = 6 %TO &CNT. ;
          #4  @26 "You have messages in your log." Color  = BLUE
          #&_I_ @26 "&&CNTERR&_I_." color = RED
      %END;
   %END;
 %MEND RPT;

 DATA _NULL_;
      %WINDOW MSG1
          %RPT()
       #13 @26 'Press ENTER to exit' Color  = BLUE;
       %DISPLAY MSG1;
```

The text file created earlier by the macro is then deleted using the Windows X command.

```
X DEL "&INDAT.2.LOG";
```

Figure 1 shows the screen that alerts the SAS programmer of unwanted messages in the log.

Figure 1



Conclusion

The application presented in this paper alerts a SAS programmer to possible unwanted messages in the SAS log and helps with the debugging process. This macro works only when the SAS program is submitted in batch mode and was developed on SAS v9.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author:

Ronald R. Palanca – RPalanca@mathematica-mpr.com

Mathematica Policy Research
600 Alexander Park
Princeton, NJ 08543
(609) 799-3535

Appendix: SAS Program

```
%MACRO  MPR_CHECKLOG(INDAT);
OPTIONS NOSYNTAXCHECK NOXWAIT NOXSYNC NOMPRINT OBS=MAX NONOTES;

**Copy preliminary log**;
X "COPY &INDAT..LOG &INDAT.2.LOG";
DATA _NULL_;
  X=SLEEP(3);
RUN;

FILENAME INDATA "&INDAT.2.LOG" ;

**Make a dataset out of preliminary log**;
DATA CHECKLOG;
  LENGTH ROWS $200 ;
  LABEL ROWS = 'Data from LOG' ;
  INFILE INDATA END=END TRUNCOVER ;
  INPUT ROWS & ;
  FLGERROR = 0;
  IF SUBSTR(ROWS,1,5) = 'ERROR' OR SUBSTR(ROWS,1,7) = 'WARNING' OR
     INDEX(UPCASE(ROWS),"UNINITIALIZED") > 0  OR INDEX(UPCASE(ROWS),"_ERROR_") > 0 OR
     INDEX(UPCASE(ROWS),"REPEATS OF BY VALUES") > 0 OR
     INDEX(UPCASE(ROWS),"EXTRANEOUS") > 0 OR
     INDEX(UPCASE(ROWS),"INVALID DATA FOR") > 0  OR
     INDEX(UPCASE(ROWS),"SAS SYSTEM STOPPED PROCESSING") > 0 OR
     INDEX(UPCASE(ROWS),"INVALID ARGUMENT") > 0 OR
     INDEX(UPCASE(ROWS),"ODS PDF PRINTED NO OUTPUT") THEN OUTPUT;
RUN;

***** Get rid of quotation marks that may affect the output ******;
DATA CHECKLOG;
   SET CHECKLOG;
   ROWS = TRANSLATE(ROWS,' ',"", " ","");
RUN;

***Delete duplicates and assign a number to each row ***;
PROC SORT NODUPKEY; BY ROWS;

DATA CHECKLOG;
  SET CHECKLOG;
  COUNT + 0;
  BY ROWS;
  IF FIRST.ROWS THEN COUNT = COUNT + 1;
RUN;

PROC SORT; BY FLGERROR COUNT;

***Check if dataset has observations ***;
PROC SQL NOPRINT;
  SELECT COUNT(*) INTO: NUMOBS
  FROM CHECKLOG
  QUIT;

***If dataset checklog has observations **;
%IF &NUMOBS GE 1 %THEN %DO;
   ** Create macro variables out of some variables **;
   DATA _NULL_;
   SET CHECKLOG END=LAST;
   IF LAST THEN DO;
```

```
      CALL SYMPUT('CNT', PUT(COUNT + 5, 3.));
    END;
    RUN;

    DATA _NULL_;
    SET CHECKLOG ;
      CALL SYMPUT('CNTERR'||LEFT(_N_ + 5), ROWS);
    RUN;

    %MACRO RPT();
      %IF &NUMOBS. GE 5 %THEN %DO;
        %DO _I_ = 6 %TO 10 ;
            #4  @26 "You have messages in your log." Color  = BLUE
            #&_I_ @26 "&&CNTERR&_I_." color = RED
        %END;
      %END;
      %ELSE %IF &NUMOBS. < 5 %THEN %DO;
        %DO _I_ = 6 %TO &CNT. ;
            #4  @26 "You have messages in your log." Color  = BLUE
            #&_I_ @26 "&&CNTERR&_I_." color = RED
        %END;
      %END;
    %MEND RPT;

    **Output the unwanted messages**;
    DATA _NULL_;
       %WINDOW MSG1
            %RPT()
            #13 @26 'Press ENTER to exit' Color  = BLUE;
        %DISPLAY MSG1;
  %END;

  **Delete the copy of preliminary log** ;
  X DEL "&INDAT.2.LOG";
  FILENAME INDATA ;

%MEND MPR_CHECKLOG;

****** How to call – add at the end of SAS program ****;
**OPTIONS SASAUTOS=('.') NOSYMBOLGEN ;
**%MPR_CHECKLOG(NameofProgram);
```