A Mass Symphony: Directing the Program Logs, Lists, and Outputs

Tom Santopoli, Octagon Research Solutions, Inc., Wayne, PA

ABSTRACT

When executing programs in SAS®, it is efficient to direct the development versions of log, list, and output files to a different location than the production versions. Development versions are typically executed in interactive mode. However, to ensure quality, production versions are most effectively executed in batch mode. There are innovative SAS programming techniques that can automatically direct log, list, and output files to various destinations based on whether the program is executing in interactive mode (development) or in batch mode (production). These destinations can be set dynamically, according to the relative location of the executable program in the directory structure. Additionally, it is often useful for logs and lists to be displayed in the interactive windows as copies are simultaneously directed to external files. This presentation will explain an integrated approach to applying these techniques.

INTRODUCTION

To direct the log, list, and output files to appropriate locations, there are three macros that get called at various parts of the program: %MLSTLOG, %MRTFSET, and %MENDPROG.

At the very beginning of the program, %MLSTLOG is called. This macro dynamically retrieves the program name and location, and stores them in global macro variables. Another global macro variable is assigned according to whether the program is executing in interactive mode or in batch mode. If the program is executing in batch mode, then a PROC PRINTTO is used to direct the log and list files to their production folders.

At the very end of the program, %MENDPROG is called. If the program is executing in interactive mode, this macro will use the display manager to direct copies of the log and list files to their development folders, as they are displayed in the interactive windows.

If the output is a table, listing, or figure (TLF), then %MRTFSET is called in the middle of the program. This macro uses the output delivery system (ODS) to direct the output to the production folder if the program is executing in batch mode, or the development folder if the program is executing in interactive mode.

HOW IT WORKS

Every program begins with a call to the %MLSTLOG macro:

```
%macro MLSTLOG;

%global progname batchec progloc;

/*Interactive Session*/

%if %length(%sysfunc(getoption(sysin)))=0 %then %do;

     %let batchec=;
     %let execpath=%sysget(sas_execfilepath);
            %let
       progname=%lowcase(%scan(%scan(&execpath,%eval(%sysfunc(count(&execpath,\)) +
       1),\),1,. ));
      %let progloc= %substr( &execpath,1,%eval(%length(%sysfunc(tranwrd(&execpath,
      %scan(&execpath,%eval(%sysfunc(count(&execpath,\)) + 1),\),) )))-1 );

%end;

/*Batch Session*/

%if %length(%sysfunc(getoption(sysin)))>0 %then %do;

     %let batchec=Y;
     %let execpath=%sysfunc(getoption(sysin));
            %let
       progname=%lowcase(%scan(%scan(&execpath,%eval(%sysfunc(count(&execpath,\)) +
       1),\),1,. ));
      %let progloc= %substr( &execpath,1,%eval(%length(%sysfunc(tranwrd(&execpath,
      %scan(&execpath,%eval(%sysfunc(count(&execpath,\)) + 1),\),) )))-1 );

     proc printto log="&progloc\..\..\logs\&progname..log"
                  print="&progloc\..\..\lst\&progname..lst"
                  new;
     run;

%end;

%mend MLSTLOG;
```

For the purpose of this illustration, we will use a program named ‘t-01.sas’ that resides in the following path:

H:\Client\HealthMiracles\ISS\Programming\Programs\Tables

The first function of %MLSTLOG is to determine if the program is executing in interactive mode or in batch mode. If the program is executing in batch mode, then the system option SYSIN will contain the program path\name:

H:\Client\HealthMiracles\ISS\Programming\Programs\Tables\t-01.sas

Therefore, the following statement will be true:

```
%length(%sysfunc(getoption(sysin)))>0
```

The GETOPTION function is used to obtain the value of the system option SYSIN. SYSFUNC allows us to apply the function on the macro level. If a value is obtained (the program is executing in batch mode), then the length will be greater than zero. If no value is obtained (the program is executing in interactive mode), then the length will be zero. The global macro variable BATCHEC is assigned a value of ‘Y’ if the program is executing in batch mode, and a null value if the program is executing in interactive mode.

Next, we must assign the program path\name to the global macro variable EXECPATH, which will be used to set the program name and location. If the program is executing in batch mode, then the path\name is retrieved as follows:

```
%let execpath=%sysfunc(getoption(sysin));
```

If the program is executing in interactive mode, then the program path\name is contained in the environmental variable SAS_EXECFILEPATH, and can be retrieved using a similar approach with the SYSGET macro function:

```
%let execpath=%sysget(sas_execfilepath);
```

In both cases, the global macro variable EXECPATH will have the value:

```
H:\Client\HealthMiracles\ISS\Programming\Programs\Tables\t-01.sas
```

The program name is extracted and assigned to the global macro variable PROGNAME using the following algorithm:

```
%let progname=%lowcase(%scan(%scan(&execpath,%eval(%sysfunc(count(&execpath,\)) +
1),\),1,. ));
```

The COUNT function is used to count the number of backslashes '\' and the EVAL function adds 1 to determine the number of the word that represents the program name, where the backslash is the delimiter. The SCAN function is then applied to obtain the program name. The SCAN function is applied again to remove the .SAS extension.

Similarly, the following algorithm is used to assign the program location to the global macro variable PROGLOC:

```
%let progloc=%substr(&execpath,1,%eval(%length(%sysfunc(tranwrd(&execpath,
%scan(&execpath,%eval(%sysfunc(count(&execpath,\)) + 1),\),) )))-1 );
```

The program name, including the .SAS extension, is extracted as explained previously. The TRANWRD function is used to remove it from the string represented by the global macro variable EXECPATH. At that point, we are left with the string:

```
H:\Client\HealthMiracles\ISS\Programming\Programs\Tables\
```

To remove the final backslash, the length of the path name is determined by using the EVAL function to subtract 1 from the result of the LENGTH function. The length of the path name can then be used in the SUBSTR function to extract the path name.

If the program is executing in batch mode, the log and list files are directed to the production folders with a PROC PRINTTO:

```
proc printto log="&progloc\..\..\logs\&progname..log"
             print="&progloc\..\..\lst\&progname..lst"
             new;
run;
```

The global macro variable PROGLOC is used to specify the location of the executable program. The convention '\..' tells SAS to move up one level in the directory structure. In a typical directory structure (Figure 1), the program folder is two levels above the Tables, Listings, and Figures folders, which is where those respective TLF programs reside. The Log, List, and Output folders contain DEV subfolders for the development versions of the Log, List, and Output files.

The global macro variable PROGNAME designates the names of the log and list files, with the .LOG and .LST extensions, respectively.

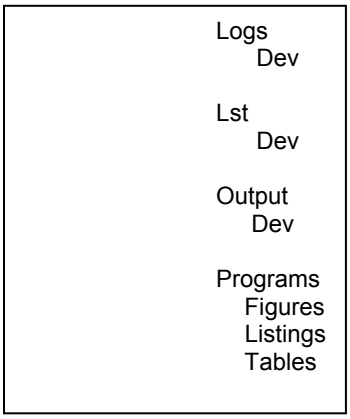

Figure 1.

If the program is executing in interactive mode, then the %MENDPROG macro directs the log and list files to the development folders:

```
%macro MENDPROG;

%if &batchec ne Y %then %do;

             dm log 'log; file  "&progloc\..\..\logs\dev\&progname..log" replace';
             dm output 'output; file  "&progloc\..\..\lst\dev\&progname..lst" replace';
%end;

%mend MENDPROG;
```

The display manager (DM) directs copies of the log and list files to the development subfolders as they are displayed in the interactive windows. This macro must be placed at the end of the program because DM can only capture logs and lists that are already displayed in the interactive windows. Therefore, the code must have already executed. If the same program or another program is executed interactively in the same SAS session, then the interactive windows should be cleared, or the new log and list files will be appended to the log and list files from the previous execution.

If the executing program is a TLF, then the %MRTFSET macro will be called to direct the output to the appropriate production or development subfolder:

```
%macro MRTFSET;

%if &batchec=Y %then %do;
      ods rtf file="&progloc\..\..\Output\%cmpres(%lowcase(&progname.)).rtf";
%end;

%else %do;
      ods rtf file="&progloc\..\..\Output\Dev\%cmpres(%lowcase(&progname.)).rtf";
%end;

%mend MRTFSET;
```

PROCESS IMPROVEMENTS

There are several benefits to directing the log, list, and output files using the above approach. When a PROC PRINTTO is opened at the beginning of a program, it must be closed at the end. The program specific name and path must also be included in the opening PROC PRINTTO. Figure 2 illustrates the outline of a very simple program:

```
proc printto
   log=" H:\Client\HealthMiracles\ISS\Programming\Programs\Tables\t-01.sas
..log"
   print=" H:\Client\HealthMiracles\ISS\Programming\Programs\Tables\t-01.sas
..lst"
    new;
run;

%include "H:\Client\HealthMiracles\ISS\Programming\Programs\Maclib\startup.sas"

*****************************************************
Program Code
*****************************************************

proc printto;
run;
```

Figure 2.

With the new approach, the above code is replaced with simple calls to %MLSTLOG and %MENDPROG (Figure 3):

```
%MLSTLOG;

%include "&progloc\..\Maclib\startup.sas";

*****************************************************
Program Code
*****************************************************

%MENDPROG;
```

Figure 3.

A typical STARTUP script that is included within the program can also make use of '\..' and the dynamically set path contained in the global macro variable PROGLOC to set libnames. This is especially useful when programs are copied to a new study in a new directory. The path no longer has to be changed to prevent outputs in the previous directory from being overwritten.

When programmers troubleshoot programs during development, it is desirable for the logs and lists to be conveniently displayed in the interactive windows for inspection. Therefore, the programmer faces the nuisance of commenting out the opening and closing PROC PRINTTO, and then uncommenting the opening and closing PROC PRINTTO at the end of the development session. Often, programmers may neglect to uncomment the PROC PRINTTO. As a result, the most current versions of the log files may not be available for examination by a standard logcheck macro that is usually applied later. If a log file for a given program does not show up as clean, an investigation will ensue. Even though the issue may really not exist in the current version of the program, time and resources may be wasted. The new approach alleviates these concerns.

When a modification to a program is made, the resulting TLF would normally overwrite the old TLF. With the new approach, if the program is executed in interactive mode, the %MRTFSET macro will direct the new TLF to the DEV subfolder. The old version of the TLF will remain intact in the main production folder until the program is submitted in batch mode. This allows an immediate comparison without copying the old TLF to an archive folder. Also, when a program is executed in interactive mode, the resulting TLF is automatically

displayed in SAS viewer. If the TLF in the production folder is opened, then the old and new versions can be conveniently eye-balled side-by-side to confirm the expected changes, before the program is submitted for production in batch mode (Figure 4).

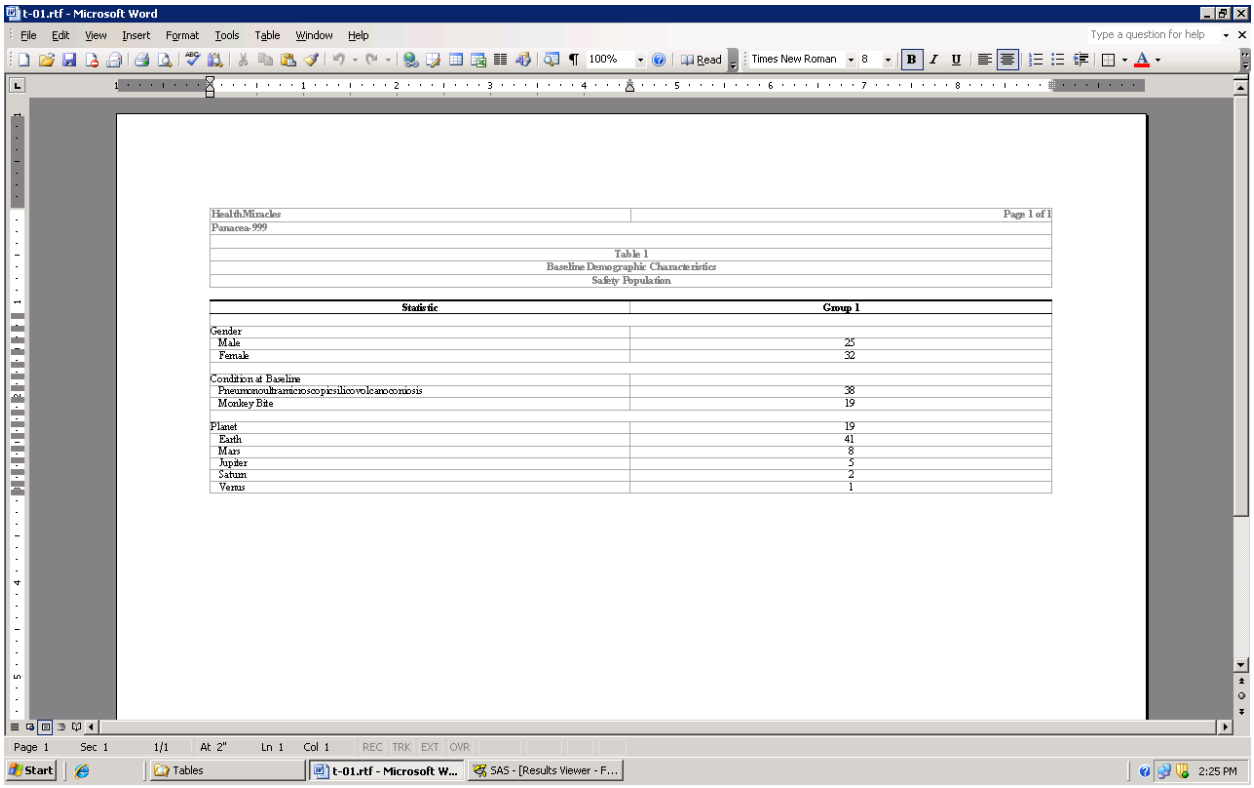

Figure 4.

If the expected results are not obtained, then permanent copies of the log and list files are available for comparison to help trace the issue.

CONCLUSION

Overall, directing the log, list, and output files using this approach helps to make the code simpler, more reusable, and more dynamic. Greater ability to conveniently examine and compare log, list, and output files should result in higher quality outputs. To ensure that programs are executed in a standard environment, it is a common standard operating procedure to execute production runs in batch mode. The methods described in this presentation draw a distinction between production runs and development runs, and could thus help to enforce this standard operating procedure.

ACKNOWLEDGMENTS

I would like to thank Allan Glaser, Terek Peterson, and Brian Shilling for their assistance and direction during the development and implementation of this approach.

CONTACT INFORMATION

Tom Santopoli
Octagon Research Solutions, Inc.
585 East Swedesford Road
Wayne, PA 19087
tsantopoli@octagonresearch.com