PharmaSUG 2022 – Paper QT144

Smart Batch Run your SAS® Programs

Wayne Zhong, Accretion Softworks

ABSTRACT

While batch running your SAS programs, you may need more features than simply running all programs. This paper shows how you can: redirect the log for each program, clean temporary files between runs, summarize errors, and pass environmental macro variables into each executing program.

INTRODUCTION

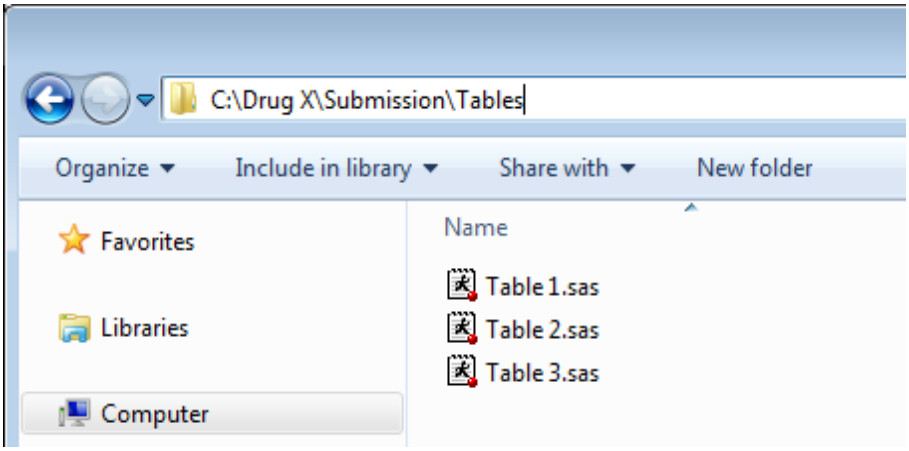
A common task encountered while leading a SAS programming project is to run all the programs in a folder. The reasons might be a newer cut of data is available, the earlier of a dependent sequence of programs was updated, or a full accounting of the status of all programs is needed.

In addition to running all programs, it is a time saver and sometimes necessary to incorporate additional tasks into the process. For example, temporary files need to be cleaned between each program run, logs from each programs can be directed into a separate folder and checked for errors, or a title can be passed to outputs.

This paper introduces a simple process to execute all the SAS files in a folder and insert additional SAS code to run alongside each program.

LIST OF FILES

The first step to batch running all files in a folder is reading the list of files from a folder. Suppose there are 3 table programs in the following folder:



The following code uses built in SAS functions to create a dataset called FILES using the pre-specified folder path.

```
%let folder=C:\Drug X\Submission\Tables

data files;
  length name $200 dir $2000;
  dir="&folder";
  did = filename('folder',dir);
```

```
  did = dopen('folder');
  do i = 1 to dnum(did);
    name = dread(did,i);
    if index(upcase(name),'.SAS') then output;
  end;
  did = dclose(did);
  keep name dir;
run;
```

The dataset FILES contains both the folder path and all filenames containing the string ".sas".

VIEWTABLE: Work.Files

	name	dir
1	Table 1.sas	C:\Drug X\Submission\Tables
2	Table 2.sas	C:\Drug X\Submission\Tables
3	Table 3.sas	C:\Drug X\Submission\Tables

BATCH RUN

Batch running all the files can be done using the CALL EXECUTE function on the FILES dataset. This function takes a string and runs it as SAS code. Here, the provided DIR and NAME variable values are used to create a **%include** statement for each program name, and the **%nrstr** wrapper is added to delay the execution of the code until after the null DATA STEP completes as otherwise it will execute immediately. The _NULL_ dataset name indicates that no dataset is created from this step.

```
options source source2;

data _null_;
  set files;
  call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
run;
```

The log from running this step is as follows. Please note that after the null DATA STEP, each program from the folder is executed in sequence using an %include statement.

```
35
36   options source source2 nomprint nospool;
37
38   data _null_;
39     set files;
40     call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
41   run;

NOTE: There were 3 observations read from the data set WORK.FILES.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds


NOTE: CALL EXECUTE generated line.
1    + %inc "C:\Drug X\Submission\Tables\Table 1.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 1.sas is file C:\Drug
      X\Submission\Tables\Table 1.sas.
42   +** Running code for Table1.sas **;
NOTE: %INCLUDE (level 1) ending.
2    + %inc "C:\Drug X\Submission\Tables\Table 2.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 2.sas is file C:\Drug
      X\Submission\Tables\Table 2.sas.
43   +** Running code for Table2.sas **;
NOTE: %INCLUDE (level 1) ending.
3    + %inc "C:\Drug X\Submission\Tables\Table 3.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 3.sas is file C:\Drug
      X\Submission\Tables\Table 3.sas.
44   +** Running code for Table3.sas **;
NOTE: %INCLUDE (level 1) ending.
```

MACRO VARIABLES

It is possible to pass macro variable values into each program as it is run. Below, the macro variables STATUS and PRGNAME are passed into each program. STATUS is static for each program, whereas PRGNAME will be provided the filename for each file.

```
options source source2 symbolgen;
%let status=DRAFT;
data _null_;
  set files;
  call execute('%nrstr(%let prgname='||strip(name)||';)');
  call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
run;
```

The **symbolgen** option shows in the log that the title for each table program is passed the correct values.

```
137
138  options source source2 symbolgen;
139  %let status=DRAFT;
140  data _null_;
141    set files;
142    call execute('%nrstr(%let prgname='||strip(name)||';)');
143    call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
144  run;

NOTE: There were 3 observations read from the data set WORK.FILES.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds


NOTE: CALL EXECUTE generated line.
1    + %let prgname=Table 1.sas;
2    + %inc "C:\Drug X\Submission\Tables\Table 1.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 1.sas is file C:\Drug
      X\Submission\Tables\Table 1.sas.
145 +** Running code for Table1.sas **;
146 +title1 j=l "Program name: &prgname" j=r "Status = &status";
SYMBOLGEN:  Macro variable PRGNAME resolves to Table 1.sas
SYMBOLGEN:  Macro variable STATUS resolves to DRAFT
NOTE: %INCLUDE (level 1) ending.
3    + %let prgname=Table 2.sas;
4    + %inc "C:\Drug X\Submission\Tables\Table 2.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 2.sas is file C:\Drug
      X\Submission\Tables\Table 2.sas.
147 +** Running code for Table2.sas **;
148 +title1 j=l "Program name: &prgname" j=r "Status = &status";
SYMBOLGEN:  Macro variable PRGNAME resolves to Table 2.sas
SYMBOLGEN:  Macro variable STATUS resolves to DRAFT
NOTE: %INCLUDE (level 1) ending.
5    + %let prgname=Table 3.sas;
6    + %inc "C:\Drug X\Submission\Tables\Table 3.sas";
NOTE: %INCLUDE (level 1) file C:\Drug X\Submission\Tables\Table 3.sas is file C:\Drug
      X\Submission\Tables\Table 3.sas.
149 +** Running code for Table3.sas **;
150 +title1 j=l "Program name: &prgname" j=r "Status = &status";
SYMBOLGEN:  Macro variable PRGNAME resolves to Table 3.sas
SYMBOLGEN:  Macro variable STATUS resolves to DRAFT
NOTE: %INCLUDE (level 1) ending.
```

CLEANUP

If it is desired to clean up temporary datasets and macro variables between executions of %include files, the following code deletes all datasets in the WORK folder and removes user created macro variables. Please note that macro variables intended to be passed through to each program should be exempted from deletion. For more details, please refer to *Bininger 2009*[1].

```
%macro cleanup();
proc datasets library = work kill nolist;
quit;

proc sql noprint;
 select name into :macvars separated by ' ' from dictionary.macros
 where scope='GLOBAL' and name not like 'SYS%' and name not in ('PRGNAME'
'STATUS');
quit;
%if %symexist(macvars) %then %symdel &macvars;

%mend cleanup;
```

Calling %cleanup after each program included is done with another CALL EXECUTE statement.

```
options source source2 symbolgen;
%let status=DRAFT;

data _null_;
  set files;
  call execute('%nrstr(%let prgname='||strip(name)||';)');
  call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
  call execute('%nrstr(%cleanup;);');
run;
```

LOG REDIRECTION

The PRINTTO procedure can redirect the log to any location, and a simple macro uses a CALL EXECUTE function to name each log file before the next SAS program is executed.

```
%macro log(name=);
proc printto new log="&folder\log\&name..log";
run;
%mend log;

data _null_;
  set files;
  call execute('%nrstr(%let prgname='||strip(name)||';)');
  call execute('%nrstr(%log(name='||strip(name)||');)');
  call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
  call execute('%nrstr(%cleanup;);');
run;

proc printto log=log;
run;
```

The NEW option in PROC PRINTTO ensures that a new log file is created each time as otherwise logs will be appended together. The final PRINTTO resets the log to the log screen. Please see *Santopoli 2010*[2] for further reading on log redirection.

Finally, the logs can be checked together for unwanted errors and warnings via a log checking macro. The details of this macro are too in-depth for this paper so the reader is directed to Watson 2017 for further reading.

```
%logcheck(dir=&folder\log);
```

The full body of the code presented in this paper can be found in Appendix A.

CONCLUSION

This paper demonstrated batch running a folder of SAS programs while passing macros variables for a title using macro variables and directing logs to a folder to simplify checking. Doing so in the batch program is simpler than editing each individual program to the same effect, making for an efficient and hopefully time saving process.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact Wayne at:

Author Name: Wayne Zhong
Company: Accretion Softworks
Email: wayne@asoftworks.com

REFERENCES

1. Chuck Bininger "Proper Housekeeping – Developing the Perfect “Maid” to Clean your SAS® Environment" < https://www.lexjansen.com/nesug/nesug09/cc/CC11.pdf>

2. Tom Santopoli "A Mass Symphony: Directing the Program Logs, Lists, and Outputs" <https://www.lexjansen.com/nesug/nesug10/bb/bb09.pdf>

3. Richann Watson " Check Please: An Automated Approach to Log Checking" <https://support.sas.com/resources/papers/proceedings17/1173-2017.pdf>

APPENDIX A

```
%let folder=C:\Drug X\Submission\Tables;

data files;
  length name $200 dir $2000;
  dir="&folder";
  did = filename('folder',dir);
  did = dopen('folder');
  do i = 1 to dnum(did);
    name = dread(did,i);
    if index(upcase(name),'.SAS') then output;
  end;
  did = dclose(did);
  keep name dir;
run;

%macro cleanup();

proc datasets library = work kill nolist;
quit;

proc sql noprint;
 select name into :macvars separated by ' ' from dictionary.macros
 where scope='GLOBAL' and name not like 'SYS%' and name not in ('PRGNAME'
'STATUS' 'FOLDER');
quit;

%if %symexist(macvars) %then %symdel &macvars;

%mend cleanup;

%macro log(name=);
proc printto new log="&folder\log\&name..log";
run;
%mend log;

data _null_;
  set files;
  call execute('%nrstr(%let prgname='||strip(name)||';)');
  call execute('%nrstr(%log(name='||strip(name)||');)');
  call execute('%nrstr(%inc "'||strip(dir)||'\'||strip(name)||'");');
```

```
  call execute('%nrstr(%cleanup;);');
run;

proc printto log=log;
run;

%logcheck(dir=&folder\log);
```