MWSUG 2018 - Paper HS-119

Automated Transfer of a Sea of SAS® Programs between Data Transfers

Xuelin Li, Eli Lilly and Company, Indianapolis, IN

Jiangang Cai, Eli Lilly and Company, Indianapolis, IN

Cindy Lee, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

In pharmaceutical industries, huge number of SAS programs are rerun routinely to refresh SDTM/ADaM data sets and TFLs (Tables, Figures, Listings) in different folder locations to accommodate data transfers. Manually updating the paths within the programs can be tedious. In this paper, we developed a method to move the SAS programs to the new location with automated path update within the programs. This method facilitates the process of moving SAS programs from one location to another location seamlessly and save hours of programmers' time in updating every program. More importantly, this automation process eliminates the chance for human errors.

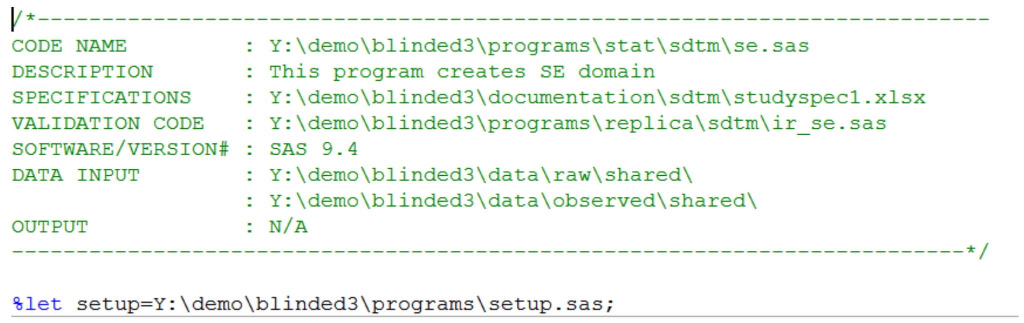
INTRODUCTION

SAS system is very flexible in term of reading and writing different types of raw files. People rarely think of SAS program files as raw data files. In this paper, the raw files are the SAS programs that are moved from one location to another location with some unique modifications, such as the setup path and the program headers.

This paper will go through the major steps that are required to achieve this goal. First, we use a data step with INFILE/INPUT statement to read a SAS program and create a temporary dataset. The lines of code from the SAS program is read into the dataset as observations With the dataset, certain characters, such as the path name and the program header, can be easily modified with a combination of SAS functions. Secondly, from the modified dataset we use a data step with FILE/PUT statements to write a new SAS program in the new location with updated path name and/or program header. For quality control, a temporary dataset is created showing lines of code that have been modified for the SAS program. Finally, if multiple SAS programs need to be modified and moved, we use the PIPE engine in the FILENAME statement to collect file names of all the SAS programs in a location and create a macro variable for the list of file names. With this macro variable, the above step can be repeated for all the SAS programs in a DO loop macro.

MODIFY A SAS PROGRAM AS A SAS DATASET

The code below shows a simplified SAS program header and the macro variable that sets the working environment. The current working folder is “Y:\demo\blinded3”. When the program is to be reused in “Y:\demo\blinded4” folder, the headers and the macro variable need to be updated. In the following sections we will go through the steps that we developed in order to achieve this goal. (Note: usually we don't need to use the macro variable to set up the working folders since the SAS sysget function can automatically recognize the program path. However, some SAS servers are supported by Linux system, in which the sysget function cannot be used.).



```
/*----------------------------------------------------------------------------
CODE NAME          : Y:\demo\blinded3\programs\stat\sdtm\se.sas
DESCRIPTION        : This program creates SE domain
SPECIFICATIONS     : Y:\demo\blinded3\documentation\sdtm\studyspec1.xlsx
VALIDATION CODE    : Y:\demo\blinded3\programs\replica\sdtm\ir_se.sas
SOFTWARE/VERSION# : SAS 9.4
DATA INPUT         : Y:\demo\blinded3\data\raw\shared\
                   : Y:\demo\blinded3\data\observed\shared\
OUTPUT             : N/A
----------------------------------------------------------------------------*/

%let setup=Y:\demo\blinded3\programs\setup.sas;
```

Figure 1. The program header and working folder that need to be updated.

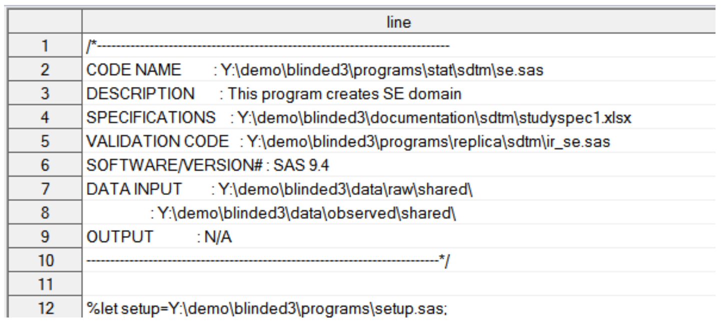
STEP 1. READ A SAS PROGRAM AS RAW DATA

First we use INFILE statement to read the SAS program. Each line of code will be read into a temporary dataset named PRE as one observation. The PRE dataset will contain one variable named line. The length of the variable is set to 1000 to avoid code truncation.

```
filename pre "Y:\demo\blinded3\se.sas";

data pre;
  infile pre lrecl=500 truncover;
  input line $char1000.;
run;

filename pre clear;
```



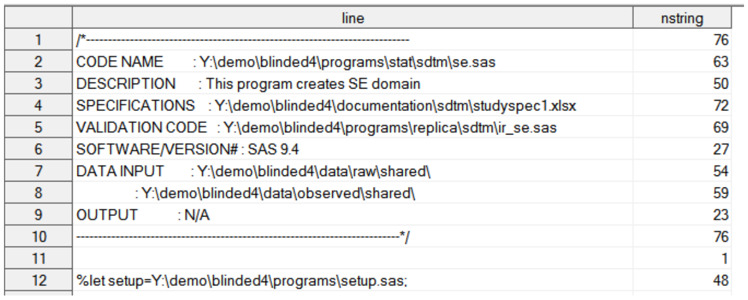
	line
1	/*--
2	CODE NAME : Y:\demo\blinded3\programs\stat\sdtm\se.sas
3	DESCRIPTION : This program creates SE domain
4	SPECIFICATIONS : Y:\demo\blinded3\documentation\sdtm\studyspec1.xlsx
5	VALIDATION CODE : Y:\demo\blinded3\programs\replica\sdtm\ir_se.sas
6	SOFTWARE/VERSION# : SAS 9.4
7	DATA INPUT : Y:\demo\blinded3\data\raw\shared\
8	: Y:\demo\blinded3\data\observed\shared\
9	OUTPUT : N/A
10	--*/
11	
12	%let setup=Y:\demo\blinded3\programs\setup.sas;

Figure 2. The SAS program is read into a SAS dataset.

STEP 2. CHANGE THE CHARACTERS IN THE SAS DATASET

Now we have the PRE dataset. The unique characters “blinded3” can be replaced by “blinded4” in a DATA step by using the TRANWRD function. A variable NSTRING is created to indicate how many characters each observation has. This variable will be used later when writing a new SAS program file. In some situations (for example replacing “blinded9” with “blinded10”) the new line of code contains more characters. Then the value of the NSTRING variable for that line need to be adjusted. The new dataset POST is created.

```
data post;
  set pre;
  file prd;
  nstring = length(trim(line));
  if index(line,"Y:\demo\blinded3") then nstring=nstring+1;
  line = tranwrd(line,"Y:\demo\blinded3","Y:\demo\blinded4");
run;
```



	line	nstring
1	/*--	76
2	CODE NAME : Y:\demo\blinded4\programs\stat\sdtm\se.sas	63
3	DESCRIPTION : This program creates SE domain	50
4	SPECIFICATIONS : Y:\demo\blinded4\documentation\sdtm\studyspec1.xlsx	72
5	VALIDATION CODE : Y:\demo\blinded4\programs\replica\sdtm\ir_se.sas	69
6	SOFTWARE/VERSION# : SAS 9.4	27
7	DATA INPUT : Y:\demo\blinded4\data\raw\shared\	54
8	: Y:\demo\blinded4\data\observed\shared\	59
9	OUTPUT : N/A	23
10	--*/	76
11		1
12	%let setup=Y:\demo\blinded4\programs\setup.sas;	48

Figure 3. Certain character in the SAS dataset is modified.

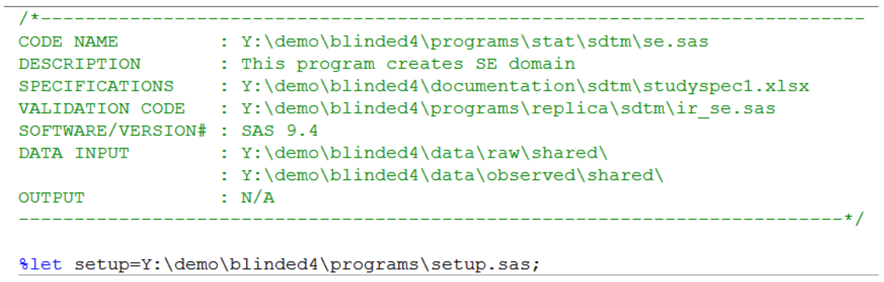
STEP 4. WRITE A NEW SAS PROGRAM IN THE NEW WORKING FOLDER

Now we have the POST dataset ready for output as a SAS program. We will use FILE statement to write the new program file in the new location. It is critical to use $varying.nstring format in the PUT statement so that there won't be trailing blanks in each line of code.

```
filename newfile "Y:\demo\blinded4\se.sas";

data _null_;
  set post;
  file newfile;
  put @1 line $varying.nstring;
run;

filename newfile clear;
```



```
/*----------------------------------------------------------------------------
CODE NAME         : Y:\demo\blinded4\programs\stat\sdtm\se.sas
DESCRIPTION       : This program creates SE domain
SPECIFICATIONS    : Y:\demo\blinded4\documentation\sdtm\studyspec1.xlsx
VALIDATION CODE   : Y:\demo\blinded4\programs\replica\sdtm\ir_se.sas
SOFTWARE/VERSION# : SAS 9.4
DATA INPUT        : Y:\demo\blinded4\data\raw\shared\
                  : Y:\demo\blinded4\data\observed\shared\
OUTPUT            : N/A
----------------------------------------------------------------------------*/

%let setup=Y:\demo\blinded4\programs\setup.sas;
```

Figure 4. A new SAS program file is created with the updated header and working path.

CONFIRM THE CORRECT MODIFICATIONS IN THE NEW PROGRAM

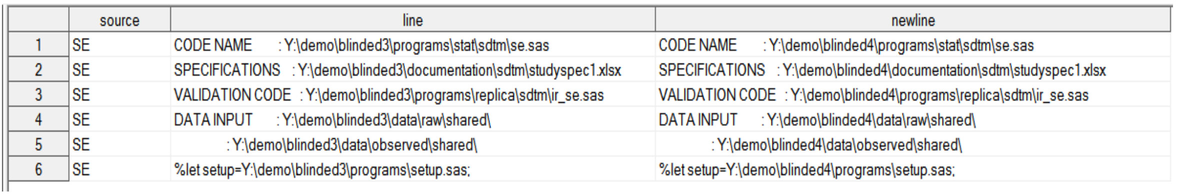
We want to make sure that the new program has the correct modifications. This comparison can be automated. We use INFILE statement to read in the new program. And then use a data step with SET statement to merge the dataset from the old program with the dataset from the new program. Only the lines of code that show difference between the two programs are kept in the new dataset COMPARE.

```
filename post "Y:\demo\blinded4\se.sas";

data newfile;
  infile post lrecl=500 truncover;
  input newline $char1000.;
run;

filename post clear;

data compare;
  length source $40;
  set pre (in=a);
  set newfile(in=b);
  source = upcase("SE");
  if line ne newline;
run;
```



	source	line	newline
1	SE	CODE NAME : Y:\demo\blinded3\programs\stat\sdtm\se.sas	CODE NAME : Y:\demo\blinded4\programs\stat\sdtm\se.sas
2	SE	SPECIFICATIONS : Y:\demo\blinded3\documentation\sdtm\studyspec1.xlsx	SPECIFICATIONS : Y:\demo\blinded4\documentation\sdtm\studyspec1.xlsx
3	SE	VALIDATION CODE : Y:\demo\blinded3\programs\replica\sdtm\ir_se.sas	VALIDATION CODE : Y:\demo\blinded4\programs\replica\sdtm\ir_se.sas
4	SE	DATA INPUT : Y:\demo\blinded3\data\raw\shared\	DATA INPUT : Y:\demo\blinded4\data\raw\shared\
5	SE	: Y:\demo\blinded3\data\observed\shared\	: Y:\demo\blinded4\data\observed\shared\
6	SE	%let setup=Y:\demo\blinded3\programs\setup.sas;	%let setup=Y:\demo\blinded4\programs\setup.sas;

Figure 5. Comparison of the SAS program before and after modifications.

AUTOMATE THE ABOVE PROCESS IF MULTIPLE PROGRAMS ARE TO BE MODIFIED

If multiple programs need to be moved to new folder, the process can be automated. First we use the PIPE engine and INFILE statement to get all the sas file names from the original folder. Then in a DATA step, we use CALL SYMPUTX statement to create macro variable for the list of file names and the number of files to be moved.

```
filename flist pipe "dir /b Y:\demo\blinded3\*.sas";

*** create a dataset for the sas files;
data rawlist;
  infile flist lrecl=200 truncover;
  input filename $100.;
  length file_name $200;
  file_name = scan(scan(filename,-1,"/"),1,".");
run;

filename flist clear;
```

```
data _null_;
  set rawlist end=last;
  length dslist $1000;
  retain dslist "";
  dslist = catx(" ",file_name,dslist);
  if last then
    do;
      call symputx("dslist",dslist);
      call symputx("n",put(_n_,best.));
    end;
run;
```

With the macro variables, we can create a macro program to automate the creation of new program files by using %DO loop. First we use the PIPE engine and INFILE statement to get all the sas file names from the original folder. Then in a DATA step, we use CALL SYMPUTX statement to create macro variable for the list of file names and the number of files to be moved.

```
%macro convert;
   %local ds;
   %do i = 1 %to &n;
     %let ds=%scan(&dslist,&i);

     filename pre "&path1/&ds..sas";

     ……

     filename newfile "&path2/&ds..sas";

     ……

   %end;
%mend;
%convert;
```

CONCLUSION

SAS system is a powerful tool that can read and write a variety types of raw data files. When it is used to read and write a SAS program file as raw file, the SAS program can be modified as a SAS dataset. This feature is especially useful when multiple SAS programs need to be moved from one location to another location with same updates in either the program header or syntax. The method described in this paper programmably creates the new SAS programs files, therefore erase the tedious work to manually update the SAS programs. It is a good example showing that a combination of some basic SAS statements and functions along with some MACRO skills can tremendously facilitate some daily job.

REFERENCES

First, Steven. 2008. “The SAS INFILE and FILE Statements”. Proceedings of the SAS Global Forum 2008 Conference. Paper 166-2008

Schacherer, Chris. 2011. “The FILENAME Statement: Interacting with the world outside of SAS®”. Proceedings of the SouthCentral SAS Users Group.

Bandi, Ranganath. 2011. “Better Ways to Speak to Your System Using SAS: Automate Routine Tasks by Using X, SYSTASK & FILENAME”. Proceedings of the SAS Global Forum 2011 Conference. Paper AD06

Bahler Caroline. 2002. “Report Creation Using Data _NULL_”. SAS Conference Proceedings: SAS User Group International. Paper 61-27

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *Pharmaceutical Statistics Using SAS®: A Practical Guide*
- *SAS® Programming in the Pharmaceutical Industry*

CONTACT INFORMATION

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

Xuelin Li
Eli Lilly and Company
317-651-5534
Xuelin.Li@lilly.com