

A Recursive SAS Macro to Automate Importing Multiple Excel Worksheets into SAS Data Sets

Wenyu Hu, Merck Sharp & Dohme Corp., Upper Gwynedd, PA
Liping Zhang, Merck Sharp & Dohme Corp., Upper Gwynedd, PA

ABSTRACT

In order to import data from Microsoft Excel into SAS[®], there are various methods such as PROC IMPORT, IMPORT WIZARD, ODBC and Excel LIBNAME engine. The optimal way often depends on the project needs and programmer's preference. The need for importing multiple worksheets adds more complexity to an already challenging job. This paper introduces a SAS macro that can automatically handle all Excel files with various numbers of worksheets in different subfolders and generate separate SAS datasets for each Excel file. As Excel files often contain multiple worksheets, we believe this macro is a flexible and user-friendly tool that has wide applicability for importing Excel files to SAS.

Keywords: proc import, Excel, recursive

INTRODUCTION

In the pharmaceutical industry, SDTM and ADaM Metadata information are usually stored in Excel spreadsheets. The current version of Excel files often need to be compared to previous versions to see what changes were made to the data definition file. Since there are no good tools to compare two files and identify the exact differences, we decided to convert the Excel files to SAS datasets and then compare the SAS datasets. Normally for different studies, the number of sheets and sheet names are different, thus using hard coding is very time consuming and error prone. An automated SAS macro would be a desirable utility tool to accomplish this task.

This paper introduces a macro which reads multiple Excel files that have multi-sheets under different folders and subfolders by using a recursive macro technique and the Excel LIBNAME engine.

The Excel LIBNAME engine was introduced in SAS version 9 and constructed a new way to connect SAS files and Microsoft Excel files. By using this new LIBNAME engine, along with other data transfer tools such as PROC EXPORT, PROC IMPORT, PROC SQL or DATA step, transferring data between Excel and SAS can be greatly simplified.

THE MACRO DESIGN

The fundamental design of this macro (%xls2sas) is to use a recursive approach. The macro flow is as follows: it first opens the root folder using DOPEN. If DOPEN returns a directory id value of greater than 0, it then uses DNUM to return the number of members in this directory. Once it gets the total number of members in the folder, it uses DREAD to get the member name and the scan function to extract the second word of the member name with the period as the delimiter. If the second word is XLS, then the Excel spreadsheet is read in using Excel LIBNAME. Once the LIBNAME statement is successfully executed, the Excel workbook, the sheets, and named ranges defined in the workbook will be available in the SAS Explorer window. If the second word resolves to null, it means the first word is considered as sub-folder. %xls2sas calls itself upon reaching sub-folders not listed in the exclusion folders list. When %xls2sas is called, it repeats the same process as the above. Upon finishing reading any file or folder, it goes up one level to its parent folder to read the next XLS file or folder. The process finishes when all Excel files and folders under the root folder are read. The key to the recursive design is to localize the macro variables so that every call to %xls2sas will have its local instance of variables. This prevents subsequent %xls2sas calls from overwriting the macro variable values of previous macro calls.

MACRO CODE STRUCTURE AND PARAMETERS

THE MACRO CODE STRUCTURE IS:

```
%macro xls2sas(folder=  
    ,subfd=Y  
    ,exclfd=  
    ,startrow=);
```

STEP #1: Declare local macro variables

```
%local _j _cnt _dsid _i _num _s_ext _s_name _filename _rc;  
  
.....  
  
%let _rc=%qsysfunc(filename(filrf,&folder));  
%if &_rc=0 %then  
%do;
```

STEP #2: Open the folders

```
%let _dsid=%sysfunc(dopen(&filrf));  
%if &_dsid>0 %then  
%do;  
  
.....
```

STEP #3: Get the number of files and folders

```
%let _num=%sysfunc(dnum(&_dsid));  
%do _i=1 %to &_num;  
  
.....
```

STEP #4: Check if the file extension is XLS. If so, open Excel workbook and read in the multi-sheets or named ranges

```
%if %upcase(&_s_ext)=XLS %then  
%do;  
  
    Libname excellib excel "&folder\&_filename";  
  
    .....  
  
    libname excellib clear;  
  
%end;
```

STEP #5: If the file extension resolves to NULL, check if the read subfolder flag is Y and also it is not in the exclusion folder list. If so, call %xls2sas to open the subfolder and Excel files under it

```

%else %if %scan(&_amp;_filename, 2, .)= and &subfd=Y and
%qsysfunc(indexw(&exclfd,&_filename))=0 %then
%do;

        %xls2sas(folder=&folder\&_filename,
                subfd=&subfd,
                exclfd=&exclfd,
                startrow=&startrow)

%end;

.....

%mend xls2sas;

```

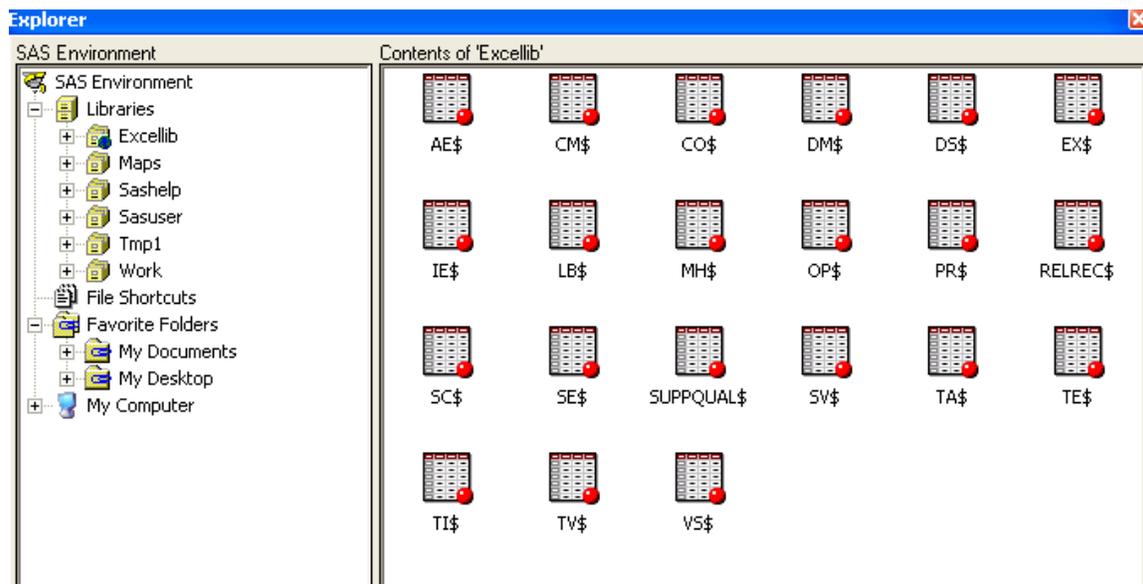
THE MACRO PARAMETERS ARE:

- folder root folder name to be searched for
- subfd include subfolder (Y/N, default=Y)
- exclfd subfolder list to be excluded
- startrow starting row of Excel worksheet

A CLOSER LOOK AT HOW TO PROCESS EXCEL WORKBOOKS

First we use the LIBNAME statement to define libref Excellib as shown in step 4 above. Once the library is established, the worksheet names become available under Excellib, as shown in figure 1 below:

Figure 1



By using Data Step or Proc SQL, we can create a list of macro variables from SASHELP.VSTABVW. In our example &sheet1, &sheet2, &sheet3, &sheet4 and &sheet5 are resolved to AE\$, CM\$, CO\$, DM\$ and DS\$ respectively. Notice that here we use SASHELP.VSTABVW instead of SASHELP.VTABLE to get sheet names because if there are quotation marks in sheet names, then SASHELP.VTABLE will not have these sheet names in memname column.

```
data _null_;
  set sashelp.vstabvw end=last;
  where libname="EXCELLIB";
  memname=upcase(scan(memname, 1, '$'));
  call symputx(cats('sheet', _n_), memname, 'L');
  if last then call symputx('_cnt', _n_, 'L');
run;

libname excellib clear;
```

After the macro variables list is created, we need to clear the LIBNAME for future use. The next step uses Proc Import combined with %do loop to read in the data from multiple Excel worksheets. In the code example below, &folder refers to the folder/subfolder %xls2sas currently processing. Since this is a recursive macro, &folder could be the subfolder under &folder user originally specified and &_filename is the Excel file currently under &folder. We also use "range" instead of the "sheet" option to specify the sheet name and combine it with the starting row if the variable values do not start on row 2. This adds more flexibility to the macro. Finally, the sheet information is appended to the master dataset created for the Excel file read in. In our example, the master dataset name is created using the folder/subfolder name together with the Excel spreadsheet name so that each Excel spreadsheet is guaranteed to have a unique master output dataset name for later comparisons.

```
%do _j=1 %to &_cnt;

  proc import datafile="&folder\&_filename"
    out=__&&sheet&_j
    dbms=excel replace;
    range="&&sheet&_j..$&startrow.:65000";
    mixed=yes;
    getnames=yes;

  run;

  .....

  data &master;
    set &master __&&sheet&_j;
  run;

  .....

%end;
```

CAPABILITIES AND LIMITATIONS WITH EXCEL LIBNAME ENGINE

The Excel LIBNAME engine has the capabilities to:

- Read data from pre-existing sheets and named ranges
- Append data to pre-existing sheets and named ranges
- Create new workbooks
- Create new spreadsheets and named ranges
- Write data to empty existing named ranges
- Delete existing data from spreadsheets
- Delete existing data from named ranges

The following are not possible via Excel LIBNAME engine

- Read data from hidden worksheets
- Delete entire workbook or spreadsheets
- Change or apply formatting, i.e, font, color
- Delete cells containing formula
- Write formulas to the cell
- Rename spreadsheets within a workbook

OTHER LIMITATIONS

- By default, SAS only scans the first 8 records of XLS file to determine column attributes/types. If the first 8 records contain only numeric data type and records with character data type are in the records afterwards, this column is considered as numeric data type when imported into SAS even though mix=yes is specified as an option in proc import statement. To solve this issue, the user needs to increase the TypeGuessRows in the Windows registry for the Microsoft Jet Engine to 0 which indicates 16,384 rows. This would force Proc Import to scan until the last non-empty data row in the workbook to determine the column data type.
- In our example of reading multiple Excel spreadsheets and appending them to the master dataset, we assumed that all worksheets within a single Excel file have similar attributes and data types because this is the nature of our metadata. Special attention is needed if the user has different data types for the same column within 2 different spreadsheets.

CONCLUSION

%xls2sas provides a flexible way to import a variable number of Excel worksheets under different folder/subfolder into SAS datasets. The macro, built with the Excel LIBNAME engine in conjunction with SASHELP.VSTABVW and recursive techniques, greatly reduces the amount of manual work.

REFERENCES

Ed Heaton "So, Your Data are in Excel!" Proceedings of the 31st Annual SAS Users Group International

Paul A. Choate, Carol A. Martell "De-Mystifying the SAS LIBNAME Engine in Microsoft Excel: A Practical Guide" Proceedings of the 31st Annual SAS Users Group International

Harry Droogendyk "Customized Excel Output Using the Excel Libname" Proceedings of South East SAS Users Group 2008

TRADEMARKS

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

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

Wenyu Hu
UG 1D-88
Merck Sharp & Dohme Corp.
Upper Gwynedd, PA 19454
(267) 305-6847
wenyu_hu@merck.com

Liping Zhang
UG 1D-10
Merck Sharp & Dohme Corp.
Upper Gwynedd, PA 19454
(267) 305-7980
liping_zhang@merck.com