Paper 034-31

Make the Invisible Visible: A Case Study of Importing Multiple Worksheet Files By Using the SAS®9 LIBNAME Engine in Microsoft Excel

Zizhong Fan, Westat, Rockville, MD

ABSTRACT

When you need to import data from Microsoft Excel into SAS®, there are various methods available. For example, you can use the IMPORT Wizard, PROC IMPORT, ODBC, PROC DBLOAD, PROC ACCESS, PROC SQL, and the most powerful, flexible, and complex tool – DDE (Dynamic Data Exchange). When multiple worksheets need to be imported, it adds more complexity to the already painful job. One of the tedious things SAS programmers have to do is to specify the different sheet names by hard coding, so that the sheet names are visible to the SAS system. Unfortunately, in most of the multi-sheet situations, the number of worksheets and sheet names happen to be different. And, sometimes the sheet names are too messy to be usable (e.g. sheet names containing quotation marks).

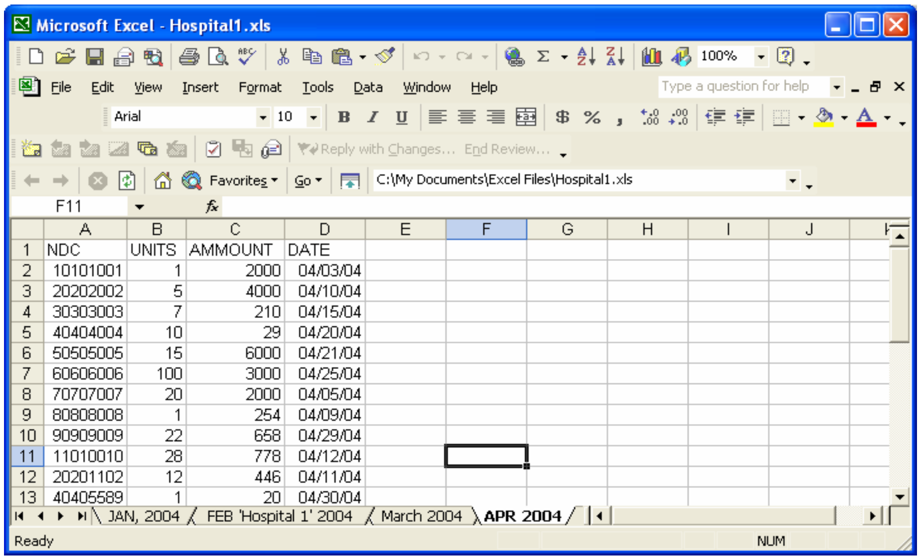
This paper provides a real–world, multi-book, multi-sheet importation situation, in which the new Excel libname engine in SAS Version 9 helped retrieve Excel work sheet names and made the complex work become a smooth ride.

INTRODUCTION

The Excel libname engine was introduced in SAS Version 9 as part of the enhancements in the SAS/ACCESS package. It constructed a new route to connect SAS files and Microsoft Excel files. By utilizing this new libname engine, along with data transfer tools such as PROC IMPORT, PROC EXPORT, DDE, or even the DATA step, transferring data between Excel and SAS can be greatly improved. This paper focuses on importing multi-sheet Excel files. The example is from the GAO (Government Accountability Office) HOPD (Hospital Outpatient Drug Data) project conducted by Westat in 2005. All of the data file names and data values have been altered for confidentiality purposes. The data values do not represent any factual meanings.

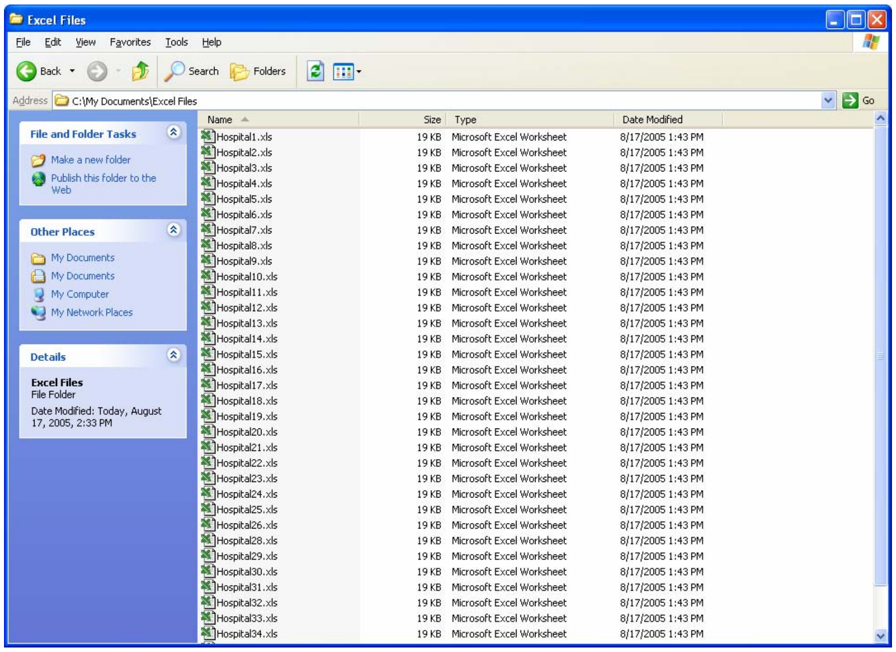
THE PROBLEM

The simplified version of the task here is to import multiple Excel files with multiple worksheets. Figure 1 shows the multiple worksheets in one of the Excel files. And Figure 2 shows all the Excel files need to be imported into SAS. In Figure 1, we can see the sheet names. But in Figure 2, the sheet names are invisible. You may wish that all the Excel files had the same number of worksheets and with the same sheet names. But unfortunately, that is not what we had. And I believe very few, if any, programmers will have such luck.



	A	B	C	D
1	NDC	UNITS	AMMOUNT	DATE
2	10101001	1	2000	04/03/04
3	20202002	5	4000	04/10/04
4	30303003	7	210	04/15/04
5	40404004	10	29	04/20/04
6	50505005	15	6000	04/21/04
7	60606006	100	3000	04/25/04
8	70707007	20	2000	04/05/04
9	80808008	1	254	04/09/04
10	90909009	22	658	04/29/04
11	11010010	28	778	04/12/04
12	20201102	12	446	04/11/04
13	40405589	1	20	04/30/04

Figure 1: Multiple worksheets with different sheet names



Address C:\My Documents\Excel Files

Excel Files
File Folder
Date Modified: Today, August 17, 2005, 2:33 PM

Name	Size	Type	Date Modified
Hospital1.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital2.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital3.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital4.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital5.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital6.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital7.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital8.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital9.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital10.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital11.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital12.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital13.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital14.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital15.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital16.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital17.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital18.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital19.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital20.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital21.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital22.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital23.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital24.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital25.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital26.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital28.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital29.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital30.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital31.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital32.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital33.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM
Hospital34.xls	19 KB	Microsoft Excel Worksheet	8/17/2005 1:43 PM

Figure 2: Multiple Excel Files

The available tools for this importation job include the IMPORT Wizard, PROC IMPORT, ODBC, PROC DBLOAD, PROC ACCESS, PROC SQL, and DDE. If we look at each of these tools, we find that we have to specify the different sheet names by hard coding them, in order to retrieve the sheet names from each workbook and pass them through a macro loop.

Here is an example of hard coding a sheet name using PROC IMPORT.

```
proc import datafile='C:\My Documents\Excel Files\Hospital1.xls'
        out= hospital1 replace;
        sheet='JAN, 2003';
        getnames=yes;
run;
```

This is an example of hard coding using DDE. Note that we need to specify variable names start and end numbers for the rows and columns, and some variable attributes.

```
filename sheet  dde "Excel|Jan, 2004!r2.c1:r5000.c4" NOTAB;

data _null_ ;
  x=sleep (5);
run;

filename mydde dde "excel|system" ;
data _null_ ;
  file mydde ;
  put "[open('C:\My Documents\Excel Files\Hospital1.xls')]";
run;

data H1200301;
  infile sheet dsd dlm='09'x truncover;
   length NDC $7 UNITS 8 AMMOUNT 8 DATE mmddyy8.;
   input  NDC UNITS AMMOUNT DATE;
run;

data _null_;
  file mydde ;
  put '[quit]';
run;
```

In order to read in all of the sheets in the Excel file, work sheet names have to be explicitly spelled out for each run. Of course, you can create your own macros to read Excel files with multiple worksheets. But, you will still need to specify the worksheet names as a parameter in the macro. So you will end up having to manually open all of the Excel files to copy the worksheet names and paste them into your programs. Now our task becomes how to make the invisible sheet names visible to SAS without doing the hard coding work. This is where the Excel libname engine in SAS Version 9 can give us a hand.

SOLUTIONS WITH THE EXCEL LIBNAME ENGINE IN SAS VERSION 9

Sample Code 1 shows how to read in all the worksheets in file Hospital1.xls and store them in SAS data sets sheet1, sheet2 …, and eventually put them into the same master SAS file. By calling the macro multiple times, we can retrieve data from the selected Excel files and put data from different sheets all into the same SAS master file regardless of how many sheets are in each Excel file or what their sheet names are.

Sample Code 1: Processing selected Excel files

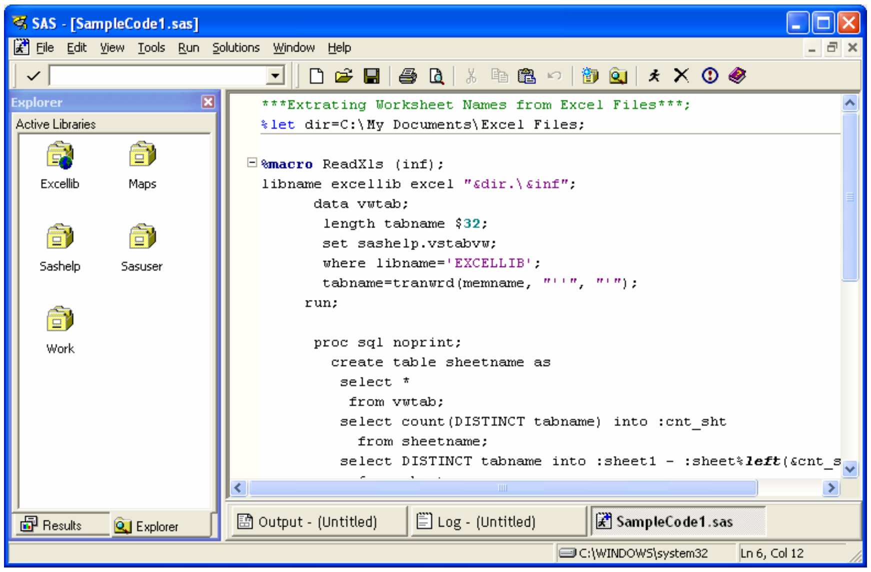
```
%let dir=C:\My Documents\Excel Files;

%macro ReadXls (inf);

libname excellib excel "&dir.\&inf";  /* STEP 1 */

proc sql noprint;                     /* STEP 2 */
        create table sheetname as
         select tranwrd(memname, "''", "'") as sheetname
          from sashelp.vstabvw
           where libname="EXCELLIB";
         select count(DISTINCT sheetname) into :cnt_sht
           from sheetname;
         select DISTINCT sheetname into :sheet1 - :sheet%left(&cnt_sht)
           from sheetname;
quit;

libname excellib clear;             /* STEP 3 */

%do i=1 %to &cnt_sht;

 proc import datafile="&dir.\&inf"  /* STEP 4 */
       out=sheet&i replace;
       sheet="&&sheet&i";
       getnames=yes;
       mixed=yes;
 run;

proc append base=master data=sheet&i force; /* STEP 5 */
   run;
%end;
%mend ReadXls;

%ReadXls (Hospital1.xls)
%ReadXls (Hospital2.xls)
%ReadXls (Hospital3.xls)
 …
```

Let's take a closer look at Sample Code 1:

- **Step 1**: First we use the libname statement to define libref excellib. When the macro variable &dir resolves, the libname statement becomes:

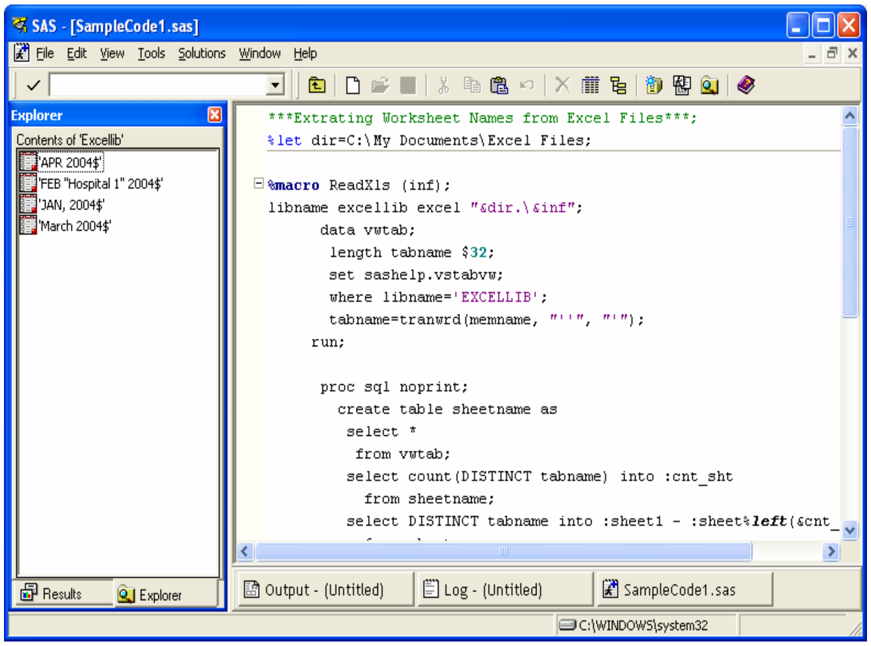
  libname excellib excel "C:\My Documents\Excel Files\Hospital1.xls";

Here *excellib* is the libref. The word *excel* is the engine name, which constructs a direct access to the Excel file. At this point, if we look at the SAS Explore window, we can see a library folder Excellib (Figure 3). If we open it, we can see worksheets are listed as separate data views (Figure 4).



```
***Extrating Worksheet Names from Excel Files***;
%let dir=C:\My Documents\Excel Files;

%macro ReadXls (inf);
libname excellib excel "&dir.\&inf";
      data vwtab;
       length tabname $32;
       set sashelp.vstabvw;
       where libname='EXCELLIB';
       tabname=tranwrd(memname, "''", "'");
     run;

      proc sql noprint;
        create table sheetname as
         select *
          from vwtab;
         select count(DISTINCT tabname) into :cnt_sht
           from sheetname;
         select DISTINCT tabname into :sheet1 - :sheet%left(&cnt_s
```

Figure 3: Excel library

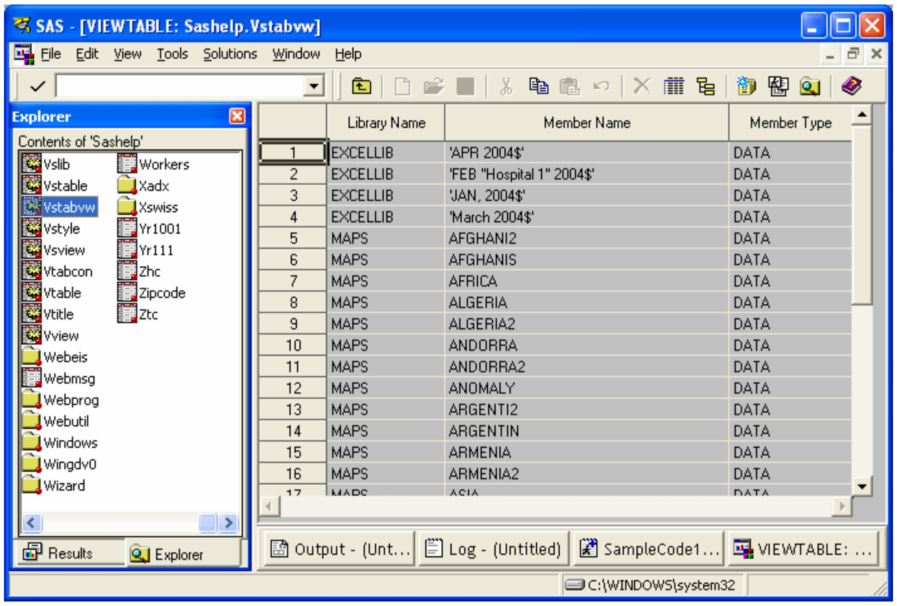


```
***Extrating Worksheet Names from Excel Files***;
%let dir=C:\My Documents\Excel Files;

%macro ReadXls (inf);
libname excellib excel "&dir.\&inf";
      data vwtab;
       length tabname $32;
       set sashelp.vstabvw;
       where libname='EXCELLIB';
       tabname=tranwrd(memname, "''", "'");
     run;

      proc sql noprint;
        create table sheetname as
         select *
          from vwtab;
         select count(DISTINCT tabname) into :cnt_sht
           from sheetname;
         select DISTINCT tabname into :sheet1 - :sheet%left(&cnt_
```

Figure 4: Work sheets in the Excel library

- **Step 2**: Once the library is established, the worksheet names become values of variable *memname* in the SASHELP dictionary view VSTABVW (Figure 5). By using PROC SQL, we create a list of macro variables from SASHELP view VSTABVW. Now, the sheet names are values for &sheet1, &sheet2, &sheet3 and &sheet4 correspondingly resolved to be `'APR 2004$'`, `'FEB ''HOSPITAL 1'' 2004$'`, `'JAN, 2004$'` and `'March 2004$'`.

  Please note that the TRANWRD function is used here to convert two consecutive single quotation marks to be one single quotation mark. This is a critical step needed to deal with single quotation marks in sheet names. It is necessary because one quotation mark is converted to two consecutive single quotation marks in SASHELP dictionary view VSTABVW. They must be converted back to a single quotation for correct reading. For example, in the SASHELP dictionary view VSTABVW, the value of memname for sheet `FEB 'HOSPITAL 1' 2004` is `'FEB ''HOSPITAL 1'' 2004$'`. It is changed back to be `'FEB 'HOSPITAL 1' 2004$'` as the value for macro variable &sheet2. In Step 4, PROC IMPORT is smart enough to read it in.



	Library Name	Member Name	Member Type
1	EXCELLIB	'APR 2004$'	DATA
2	EXCELLIB	'FEB "Hospital 1" 2004$'	DATA
3	EXCELLIB	'JAN, 2004$'	DATA
4	EXCELLIB	'March 2004$'	DATA
5	MAPS	AFGHANI2	DATA
6	MAPS	AFGHANIS	DATA
7	MAPS	AFRICA	DATA
8	MAPS	ALGERIA	DATA
9	MAPS	ALGERIA2	DATA
10	MAPS	ANDORRA	DATA
11	MAPS	ANDORRA2	DATA
12	MAPS	ANOMALY	DATA
13	MAPS	ARGENTI2	DATA
14	MAPS	ARGENTIN	DATA
15	MAPS	ARMENIA	DATA
16	MAPS	ARMENIA2	DATA
17	MAPS	ASIA	DATA

Figure 5: SASHELP dictionary view VSTABVW

- **Step 3**: Now we clear the libname. This is necessary to free up the Excel file for future use.

- **Step 4**: PROC IMPORT is used to read in the data from multiple sheets by using the %do loop. This section could be replaced by your own favorite procedure such as DDE. The benefit of using PROC IMPORT in this case is that you don't need to specify the variable names, attributes, start and end numbers for the columns and rows, and informats. PROC IMPORT does a good job of preserving formats from Excel to SAS.

  Please note that the new *mixed=yes* option is used here to avoid losing mixed data (numeric and character in the same column). This is a new enhancement in SAS Version 9. The following link has the full reference http://support.sas.com/techsup/unotes/SN/006/006123.html

- **Step 5**: Data are appended onto the master data set.

Sample Code 2 shows a more automated process to import all of the data in multiple sheets, from multiple Excel files, into the master data set in only one macro call.

Sample Code 2: Processing all the Excel files

```
options noxwait;

%macro ReadXls (dir=);
%sysexec cd &dir; %sysexec dir *.xls /b/o:n > flist.txt;

data  indexfile;
 length filen $200;
 infile "&dir./flist.txt";
 input filen $;
run;
```

```
proc sql noprint;
 select count(filen) into :cntfile from _indexfile;
 %if &cntfile>=1 %then %do;
  select filen into :filen1-:filen%left(&cntfile)
   from _indexfile;
 %end;
quit;

%do i=1 %to &cntfile;
libname excellib excel "&dir.\&&filen&i";
proc sql noprint;
        create table sheetname as
         select tranwrd(memname, "''", "'") as sheetname
          from sashelp.vstabvw
           where libname="EXCELLIB";
         select count(DISTINCT sheetname) into :cnt_sht
           from sheetname;
         select DISTINCT sheetname into :sheet1 - :sheet%left(&cnt_sht)
           from sheetname;
 quit;

%do j=1 %to &cnt sht;
     proc import datafile="&dir.\&&filen&i"
          out=sheet&j replace;
          sheet="&&sheet&j";
          getnames=yes;
          mixed=yes;
     run;

       data sheet&j;
        length  excelfilename $100 _sheetname $32;
        set sheet&j;
         excelfilename="&&filen&z";
         sheetname="&&sheet&j";
       run;

    proc append base=master data=sheet&j force;
    run;

 %end;
libname excellib clear;

%end;
%mend ReadXls;

%readxls (dir=c:\my documents\excel files)
```

Sample Code 2 has the same concept, except it is more automated to read in all the Excel files in the same folder. First, all the Excel file names in the same folder are put into file flist.txt. Then, the file names (variable filen) in flist.txt are read into data set _indexfile. PROC SQL is used to put the file names into a list of macro variables: fname1 to fname&cntfile. Then, each file goes through the same process as in sample code 1 to get sheet names retrieved by using SASHELP dictionary view VSTABVW. And data are imported by the PROC IMPORT procedure. At the end, all the data are appended to the master data set. So one macro call imports all the data.

ABOUT SHEET NAMES

One of the benefits of using libname excel engine is the ability to handle messy sheet names. The sample codes provided in this paper can handle all the possible special characters and symbols in sheet names including commas and quotation marks. Both single and double quotation marks can be read correctly. Even open quotation marks in the sheet names will not be a problem. The step 2 in sample code 1 is where the sheet names get retrieved and manipulated for correct reading. So there is no need to go through all the Excel files to check and correct messy sheet names.

Please note that when extracting sheet names, the SASHELP dictionary view VSTABVW must be used. The SASHELP dictionary view VSTABBLE or SQL dictionary table TABLES may not have all the sheet names in the memname column when there are quotation marks in sheet names.

The only precaution is that sheet names usually should not exceed 29 characters even though Excel allows 31 characters. This is because when there are blank spaces in the sheet name, or if the sheet name starts with a non-SAS standard character for SAS data set name (SAS standard being letters or underscores), SAS will put a $ sign at the end of the sheet name and surround the sheet name with single quotes. This takes three spaces. In the SASHELP dictionary view VSTABVW, the memname has a length of 32.

ABOUT EXCEL FILE TYPES

The Excel libname engine can be only used for Excel (5, 97, 2000, 2002) files. CSV files are not compatible with this engine.

CONCLUSION

The Excel libname engine is a great enhancement to SAS/ACCESS software. With the conjunctional use of the SASHELP dictionary view VSTABVW, Excel worksheet names can be retrieved and used in the importation process. This eliminates the time consuming work of pre-importation, screening, and manipulation of worksheet names.

REFERENCES:

SAS online document V9.
http://support.sas.com/91doc/docMainpage.jsp

SN-006123 Importing Excel file into SAS can result in missing data
http://support.sas.com/techsup/unotes/SN/006/006123.html

You Could Look It Up: An Introduction to SASHELP Dictionary View
http://www2.sas.com/proceedings/**sugi**26/p017-26.pdf

SAS Guide to the SQL procedure: Usage and Reference, Version 6, First Edition;
SAS Institute, Cary, NC, USA

ACKNOWLEDGEMENT

I would like to thank Michael Raithel and Duke Owen for reviewing and editing this paper.

CONTACT INFORMATION

Zizhong Fan
Westat
1650 Research Boulevard
Rockville, MD 20850
(240) 314 -2486
E-mail: JamesFan@westat.com

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.