

Using SAS® Import/Export Wizards and SAS Macros To Automate Data Manipulation of External Data

Chris D Thompson, Wachovia Corporation, Winston-Salem, NC

ABSTRACT

This paper will outline one process of using the SAS system to automate data manipulation of external data. Using this process, it will not be necessary for you to physically open the external sources of your data. The process has four major components:

- Import external data
- Manipulate the data using the SAS system
- Export the data
- Use macros to automate the first 3 components

In the example discussed in this paper, the SAS system for personal computers, version 6.12 for Windows™ is utilized. Therefore, some of the point-and-click Windows features used here may not be available for IBM® Mainframe, VAX, or UNIX users, but the generated code can be applied to all platforms.

INTRODUCTION

SAS system programmers are frequently asked to manipulate data that is stored in a non-SAS database. They are then asked to return the results of those data manipulation efforts in the original data structure. Completing this request requires three steps, with a fourth step added to automate the first three steps:

1. Import external data
2. Manipulate the data using the SAS system
3. Export the data
4. Automate steps 1 through 3

The fourth step proves particularly useful in cases where multiple iterations of the same procedures are required. Use of the SAS system's Import/Export Wizards and SAS macros provide you with all of the tools you need to complete steps 1, 3 and 4. The data manipulation itself will be left to you.

THE CHALLENGE

You are presented with 20 Microsoft Excel® workbooks, each containing one segment of the data. You are asked to re-segment the data into 11 different groupings and return 11 Excel workbooks.

In order to do this using Excel, you will need to open each workbook individually, sort the data by the new groupings, highlight and copy all records for a single

grouping, then paste that data into a new workbook for that particular grouping. In all, you will be required to open 20 existing workbooks, create 11 new workbooks, and perform 220 copy and paste maneuvers. This would be unbearably tedious.

THE SOLUTION — The SAS System

You need to perform four steps in a single SAS program to re-segment the Excel data into 11 new Excel workbooks.

1. Import an Excel workbook into a SAS data set.
2. Manipulate the data by separating it into 11 segments as requested.
3. Export a SAS data set into an Excel workbooks.
4. Write macros to perform the import 20 times and the export 11 times.

IMPORT EXTERNAL DATA

SAS Version 6.12 for Windows provides us with an Import Wizard. It can be found under the File Command on the toolbar. Once selected, a series of pop-up boxes will appear, giving you the opportunity to communicate to SAS the following information.

- File format being imported (in our case, Excel, but additional options include dBASE, Lotus, Comma Separated Values, and the most flexible, Delimited Files).
- The location of the file (this pop-up box operates much like a Windows file manager).
- Any pertinent information about the file, such as whether or not column names are in the first row.
- Into which SAS library and what SAS data set name you would like the file to be placed.

After communicating all of this information to SAS, clicking the finish button in the pop-up box will successfully create your SAS data set.

Because you want to automate this process, and you need to repeat it 20 times, you will want the code that SAS generated based on the information you supplied. By recalling text in the Program Editor, you can see this code. Also note that for non-Windows programmers, this code can be entered directly. The ACCESS procedure is used, shown below.

```
PROC ACCESS DBMS=EXCEL;
CREATE WORK._IMEX._ACCESS;
PATH="C:\DATA\DATAIN\INFILE1.XLS";
WORKSHEET='SHEET1';
GETNAMES=YES;
SCANTYPE=YES;
MIXED=YES;
CREATE WORK._IMEX._VIEW;
SELECT ALL;
RUN;
DATA WORK.INFILE1;
SET WORK._IMEX_;
RUN;
```

MANIPULATE THE DATA

Once you have the data into 20 SAS data sets, you can do any required manipulations, utilizing the full power of the SAS system. In this simple example, the only manipulation required would be to append the 20 data sets, sort by the requested field, and create 11 new data sets.

EXPORT SAS DATA

Now that you have the 11 SAS data sets, you must place that information back into Excel workbooks. This is done easily with the Export Wizard, which acts similarly to the Import Wizard. Again, after going through the wizard, the code should be recalled for automation. This time, the DBLOAD procedure is used.

```
PROC DBLOAD DBMS=EXCEL DATA=OUTFILE1;
PATH="C:\DATA\DATAOUT\NEW1.XLS";
PUTNAMES=YES;
LIMIT=0;
LOAD;
RUN;
```

MACROS FOR AUTOMATION

Now that you have the code for importing the data, manipulating the data, and exporting the data, you need to write a macro that imports 20 files, manipulates the data, and then exports 11 files all in one program, with only one submit.

The first part of the macro sets up a loop to perform 20 iterations of the import code. In a temporary data set, an array is defined which contains the names of the 20 files to be imported. A CALL SYMPUT command can then be utilized to create a macro variable called "inf" which is the *i*th value of the array, where *i* is the value of loop20. At this point, you can insert the import code, replacing the file name with the macro variable "inf" and the output data set name with *tempmacrovariableloop20* (temp1, temp2, temp3, etc. based on the value of loop20).

The second part of the macro is data manipulation.

The third part of the macro sets up a loop to run the export code 11 times. Similar to the import loop, an array is created to define the values of your macro variable. The difference is that the values of the array are not file names, rather they are the values of the variable which the data is being sorted by. This allows you to use an "if statement" in your data manipulation code to choose only those records with one particular value of the sort variable. These values can then also be used for the output SAS data sets and resulting Excel files.

You are not allowed to have a data step in the form DATA ¯ovar. The first character after DATA must be valid (ampersand is not valid). In the example, this is circumvented by naming the data sets X_¯ovar. This means that if the values of the sort variable are more than 6 characters, you will have a resulting invalid data set name. You can correct this by defining a shorter length variable in your data manipulation code.

The resulting macro represents a program to import data, manipulate data, and export data with a single submit.

EXAMPLE MACRO

```
%MACRO AUTOMATE;
%DO LOOP20 = 1 %TO 20;
DATA TEMP;
ARRAY LIST20{20} $
('INFILE1','INFILE2',...,'INFILE20');
CALL SYMPUT ('INF',TRIM(LIST20(&LOOP20)));
RUN;
* insert import code here with following changes ;
* line 3 ; PATH="C:\DATA\DATAIN\&INF";
* line 11 ; DATA WORK.TEMP&LOOP20;
%END; * ends loop20;

***** PART 2 *****;
DATA ALL; SET TEMP1 TEMP2 ... TEMP20;
PROC SORT DATA=ALL; BY sortvariable;

***** PART 3 *****;
%DO LOOP11 = 1 %TO 11;
DATA TEMPOUT;
ARRAY LIST11{11} $ ('value1','value2',...,'value11');
CALL SYMPUT ('OUTF',TRIM(LIST11(&LOOP11)));
RUN;
DATA X_&OUTF; SET ALL;
IF sortvariable = "&OUTF";
* insert export code here with following changes ;
* line 1 ; PROC DBLOAD DBMS=EXCEL
DATA=X_&OUTF;
* line 2 ; PATH="C:\DATA\DATAOUT\&OUTF";
%END; * ends loop11;
%MEND AUTOMATE; * ends macro;
%AUTOMATE; * invokes macro;
RUN;
```

CONCLUSION

As you can see, one SAS program can import multiple external data files, combine and manipulate that data, and export multiple files to a variety of data formats. This is possible through the use of PROC ACCESS, PROC DBLOAD, and SAS macros. Furthermore, PROC ACCESS code and PROC DBLOAD code can be generated for you by using the import and export wizards, or it could be coded directly by programmers using a non-Windows platform.

ACKNOWLEDGMENTS

The author wishes to thank Don Stroup and Pete Saffo for their supervision and advice during the development of this paper and the projects that instigated the paper. The author also wishes to thank Michael Miller of Wake Forest University Baptist Medical Center for his invaluable mentoring. Finally, the author wishes to thank all the members of TSUG(Triad) which provided valuable feedback from the original presentation of this paper.

SAS is a registered trademark or trademark of SAS Institute, Inc. in the USA and other countries. IBM is a registered trademark or trademark of International Business Machines Corporation. ®

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

AUTHOR

Chris D Thompson, Technical Analyst, Wachovia
8 years SAS programming experience.

Wachovia Bank, NA
Attn: Chris D Thompson
Mail Code: NC-31031
301 N. Church St.
Winston-Salem, NC 27150-3099

Phone: (336)770-4503
Fax: (336)770-7037
E-Mail: chris.thompson@wachovia.com