

Proc Import Diagnostic Macro

Shan Lee

GlaxoSmithKline, David Jack Centre for Research & Development, Ware, United Kingdom



Introduction

SAS® Proc Import examines (by default) the first 8 rows of an Excel spreadsheet in order to determine whether columns are character or numeric. Unlike SAS, where a variable is either character or numeric, it is possible for a number to be formatted as text in Excel even when other numbers in the column are stored as numeric values. This could result in unexpected results when the Excel data is imported to SAS. This poster will:

1. Describe an example of when Proc Import might generate unexpected results.
2. Describe a SAS macro that provides diagnostics in the log to identify potential issues for Proc Import, such as numbers formatted as text.

Example

Suppose all of the first 8 rows of an Excel spreadsheet contain numeric data. The remaining cells after the 8th row appear to be numeric (figure 1), but row 12 contains the number '11' formatted as text rather than numeric, as can be seen when the cell is selected and the mouse is hovered over the '!' icon (figure 2):

	A	B
1	foo	
2	1	
3	2	
4	3	
5	4	
6	5	
7	6	
8	7	
9	8	
10	9	
11	10	
12	11	
13	12	
14		

Figure 1

	A	B	C	D	E	F	G
1	foo						
2	1						
3	2						
4	3						
5	4						
6	5						
7	6						
8	7						
9	8						
10	9						
11	10						
12	11						
13	12						
14							

Figure 2

```
proc import datafile = 'c:\Example.xlsx'
  dbms = excel
  out = example replace;
  mixed = yes;
run;
```

If both character and numeric values are found in the first 8 rows, then specifying 'mixed = yes' would result in a character variable being created in the SAS dataset and any numeric values would be converted to character. However, in this example, all of the values in the first 8 rows are numeric, so it is assumed that the column does not contain mixed data and the variable 'foo' is created as a numeric. Row 12 of the Excel spreadsheet contains the value '11' which has been formatted as text; SAS cannot store a character value in a numeric variable, so this value will be missing in the SAS dataset, as shown in Figures 3 and 4:

	foo
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	-

Figure 3

Column Attributes

General | Colors | Fonts

Name: foo

Label: foo

Length: 8

Format: BEST12

Inform: 12

Type: ☒ Character ☐ Numeric

Close Apply Help

Figure 4

Macro to Diagnose Potential Issues

The author of this poster has developed a SAS macro to highlight potential data issues before calling Proc Import. The macro takes advantage of the fact that SAS can use DDE to open an Excel file and run a VBA macro that has been stored within the Excel file ('Using SAS and DDE to execute VBA macros in Microsoft Excel' - Christopher A. Roper, Qualex Consulting Services, Inc., Cary, NC. USA [1]). This 'SAS macro' works in conjunction with some VBA code stored within a macro-enabled Excel workbook, which needs to be stored in a location that is accessible by anyone wishing to use the SAS macro.

The macro has the following parameters:

```
%macro procImport(dataFile =
  ,out =
  ,replace = replace
);
```

The general approach taken by the macro is:

- Write the pathname of the Excel file to a temporary CSV file, which will later be deleted.
- Use DDE to open the macro-enabled Excel workbook- note that this is not the same as the Excel file that needs to be imported.
- Use DDE to run the VBA macro stored within the macro-enabled Excel workbook.
 - The VBA macro reads the temporary CSV file to obtain the pathname of the Excel file to be imported.
 - The VBA macro saves the Excel file as a new CSV file.
 - The VBA macro closes all open workbooks.
- After the VBA macro has stopped running, the SAS macro continues to execute.
- The first row of the CSV file containing the Excel data is imported using Proc Import- this is for the purpose of determining what the variables are in the CSV file.

- Call Proc Contents for the dataset created by Proc Import; the 'out = ' option is used with Proc Contents, in order to create a dataset storing the names of all variables from the CSV file. Knowing the names of these variables, the macro can then use 'data _null_' and 'put' statements to build a temporary file that stores SAS code for reading the CSV file with 'input' statements- the SAS code will read every variable in the CSV file as a character variable, to ensure that there will be no missing values.
- Use an 'X' command to run the SAS code for reading the CSV file using input statements.
- Now use Proc Import to read the Excel file.
- The dataset generated by reading the Excel file using Proc Import can be compared with the dataset generated by reading the equivalent CSV file using input statements. The latter should have no missing values, since every value was read into a character variable, regardless of whether or not it corresponded to a number. If any data in the Excel file had not been successfully imported due to reasons discussed in this paper, then there would be an inconsistency between the 2 datasets, and the macro would generate messages in the log to highlight the issues:

ERROR: for variable foo , 12 non-missing values were detected in the Excel file, but there are 11 non-missing values in the SAS dataset created by running Proc Import on the Excel file

References

1. Using SAS and DDE to execute VBA macros in Microsoft Excel: <http://www2.sas.com/proceedings/sugi25/25/cc/25p098.pdf> - Christopher A. Roper, Qualex Consulting Services, Inc., Cary, NC. USA

Contact Information

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

Author Name: Shan Lee
Company: GlaxoSmithKline
Address: David Jack Centre for Research and Development
Park Road
Ware
Hertfordshire
SG12 0DP
United Kingdom
Work Phone: +44 (0)1992 502369
Email: shan.2.lee@gsk.com

Brand and product names are trademarks of their respective companies.