PhUSE 2013

Paper PP23

Proc Import Diagnostic Macro

Shan Lee, GlaxoSmithKline, Ware, United Kingdom

ABSTRACT

SAS® Proc Import examines (by default) the first 8 rows of an Excel spreadsheet in order to determine whether columns are character or numeric. If the majority of the first 8 cells of a column in an Excel file contain numeric values, then the column will be imported as a SAS numeric variable; if the Excel column also contains numbers that, for whatever reason, were formatted as text in Excel, then they will be set to missing in the SAS variable, which can be misleading because it is not always visibly obvious when a number has been formatted as text in Excel.

INTRODUCTION

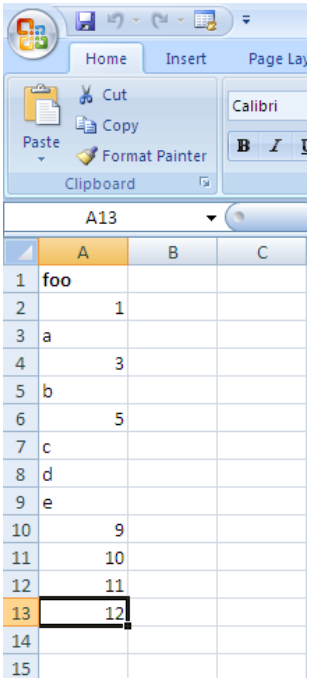
The purpose of this paper is to:

1. Discuss various scenarios in which Proc Import might generate unexpected results whilst reading numeric data from Excel spreadsheets.
2. Describe a SAS macro that provides diagnostics in the log to identify potential issues for Proc Import, such as numbers formatted as text.

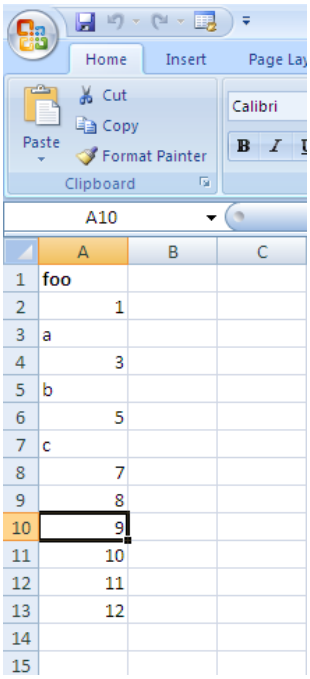
This paper will begin by presenting examples of using Proc Import to read the Excel data shown below. The examples illustrate a variety of scenarios in which SAS makes the decision on whether to read Excel data into a SAS character variable or a SAS numeric variable.

After presenting the examples, this paper will discuss how misleading results could be obtained when numbers in Excel are formatted as text. Finally, the paper will outline a SAS macro that provides diagnostics in the log to identify such issues.

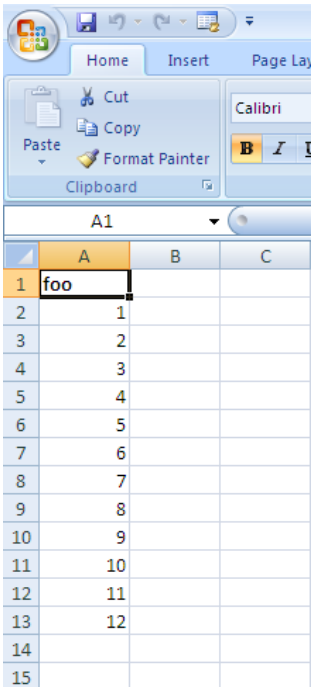
'majorityCharacter.xlsx' – the majority of the first 8 rows contain character data:



'majorityNumeric.xlsx' – the majority of the first 8 rows contain numeric data:



'allNumeric.xlsx' – all of the first 8 rows contain numeric data:

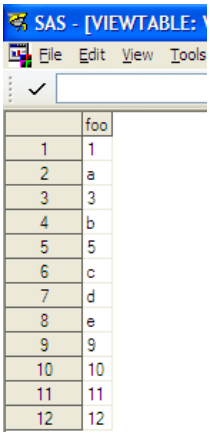


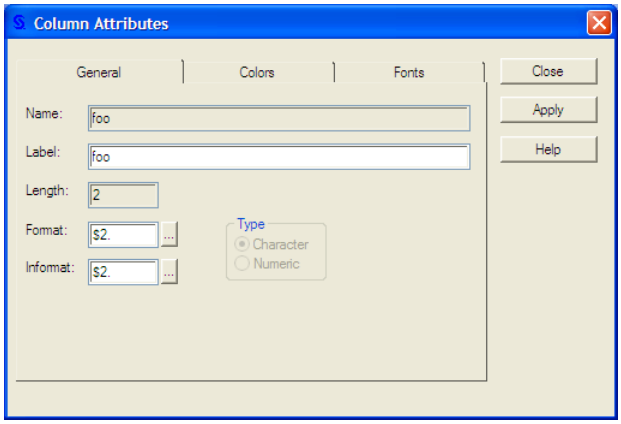
EXAMPLE 1: MAJORITY OF FIRST 8 ROWS IN EXCEL FILE CONTAIN CHARACTER DATA; USE DEFAULT 'MIXED = NO'

```
/* Majority of first 8 rows in Excel file contain character data; use default 'mixed =
no' */
proc import datafile = 'u:\PhUSE 2013\majorityCharacter.xlsx'
                    dbms = excel
```

```
                    out = test1
                    replace
                    ;
run;
```

By default, Proc Import sets 'mixed = no', so SAS determines that the variable 'foo' should be character because the majority of the first 8 rows in the Excel file contain character values. The numeric values in the Excel file are converted to character when the file is read into SAS.

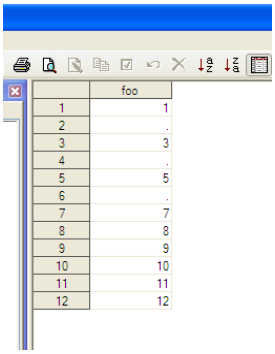


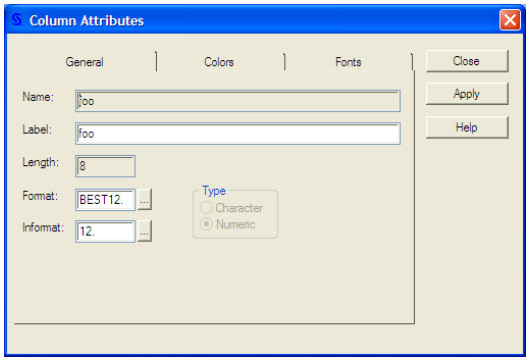


EXAMPLE 2: MAJORITY OF FIRST 8 ROWS IN EXCEL FILE CONTAIN NUMERIC DATA; USE DEFAULT 'MIXED = NO'

```
/* Majority of first 8 rows in Excel file contain numeric data; use default 'mixed =
no' */
proc import datafile = 'u:\PhUSE 2013\majorityNumeric.xlsx'
                    dbms = excel
                    out = test2
                    replace
                    ;
run;
```

By default, Proc Import sets 'mixed = no', so SAS determines that the variable 'foo' should be numeric because the majority of the first 8 rows in the Excel file contain numeric values. The character values in the Excel file cannot be converted to numbers and are set to missing when the file is read into SAS.



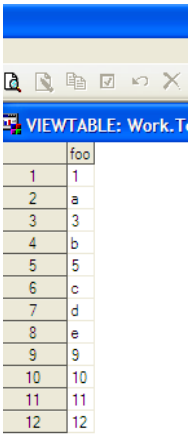


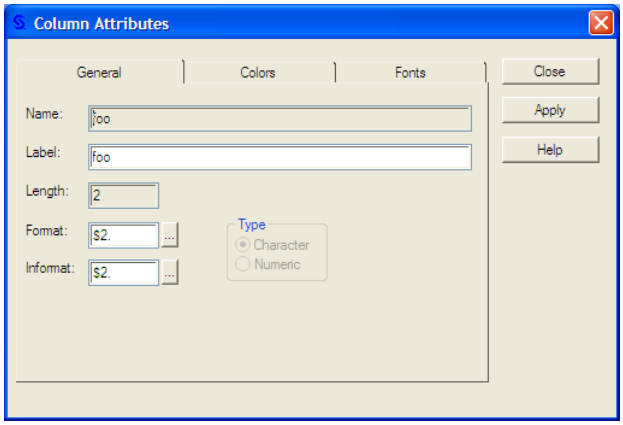
EXAMPLE 3: MAJORITY OF FIRST 8 ROWS IN EXCEL FILE CONTAIN CHARACTER DATA; USE 'MIXED = YES'

```
/* Majority of first 8 rows in Excel file contain character data; mixed = yes */
proc import datafile = 'u:\PhUSE 2013\majorityCharacter.xlsx'
                    dbms = excel
```

```
                    out = test3
                    replace
                    ;
      mixed = yes;
run;
```

In this example, 'mixed = yes', so SAS creates the variable 'foo' as character, irrespective of whether the majority of the first 8 rows in the Excel file contain character or numeric values (in this case, the majority of the first 8 rows contain character data). The numeric values in the Excel file are converted to character when the file is read into SAS.

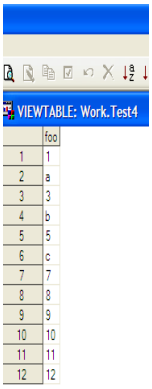


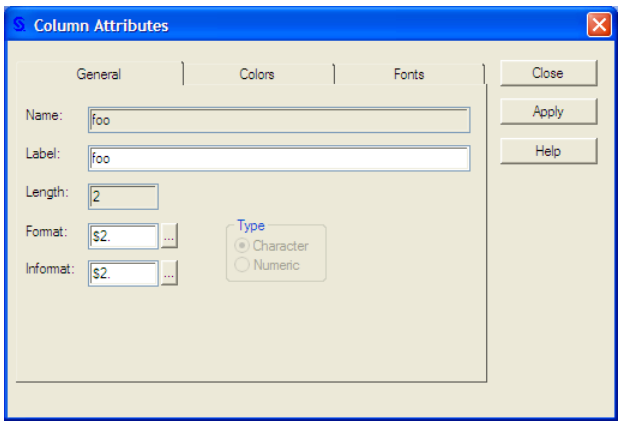


EXAMPLE 4: MAJORITY OF FIRST 8 ROWS IN EXCEL FILE CONTAIN NUMERIC DATA; USE 'MIXED = YES'

```
/* Majority of first 8 rows in Excel file contain numeric data; mixed = yes */
proc import datafile = 'u:\PhUSE 2013\majorityNumeric.xlsx'
                    dbms = excel
                    out = test4
                    replace
                    ;
      mixed = yes;
run;
```

In this example, 'mixed = yes', so SAS creates the variable 'foo' as character, irrespective of whether the majority of the first 8 rows in the Excel file contain character or numeric values (in this case, the majority of the first 8 rows contain numeric data). The numeric values in the Excel file are converted to character when the file is read into SAS.

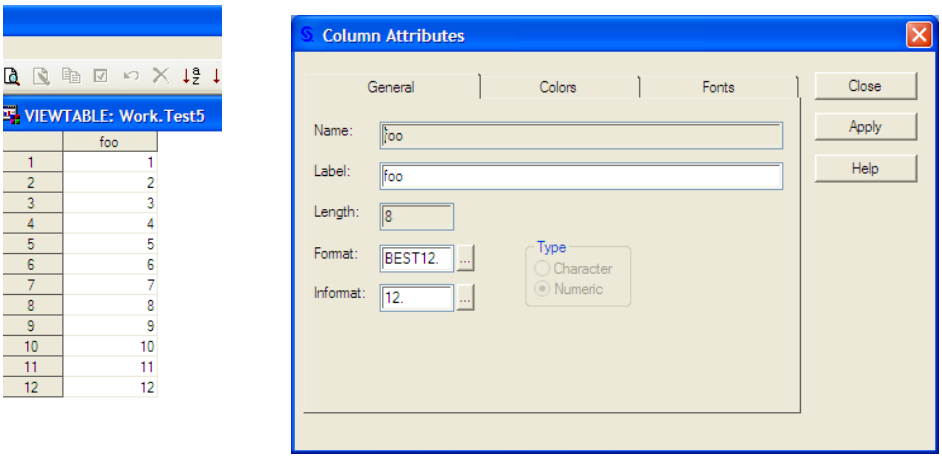




EXAMPLE 5: ALL OF THE FIRST 8 ROWS IN EXCEL FILE CONTAIN NUMERIC DATA; USE 'MIXED = YES'

```
/* All of first 8 rows in Excel file contain numeric data; mixed = yes */
proc import datafile = 'u:\PhUSE 2013\allNumeric.xlsx'
                    dbms = excel
                    out = test5
                    replace
                    ;
       mixed = yes;
run;
```

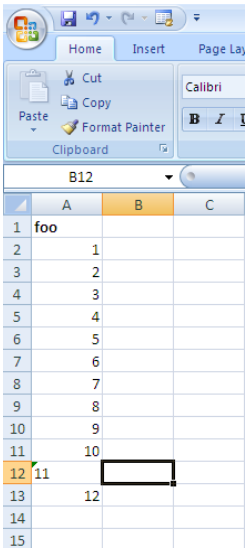
In this example, 'mixed = yes', but all of the first 8 rows contain numeric data, so SAS assumes that the column does not contain mixed data and the variable 'foo' is created as a numeric. This is fine in this case, since all the data in the Excel column is numeric and can be read into a SAS numeric variable.



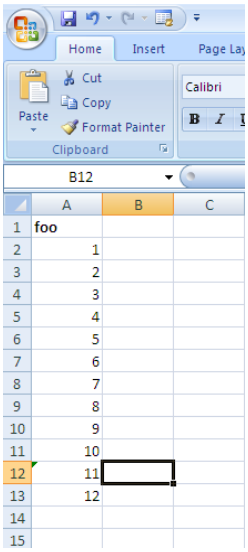
WHERE IT CAN GO WRONG

In each of the examples 2 and 5, SAS creates a numeric variable to store the Excel data. One might assume that all numbers appearing after row 8 in the Excel spreadsheet will be imported into the SAS numeric variable.

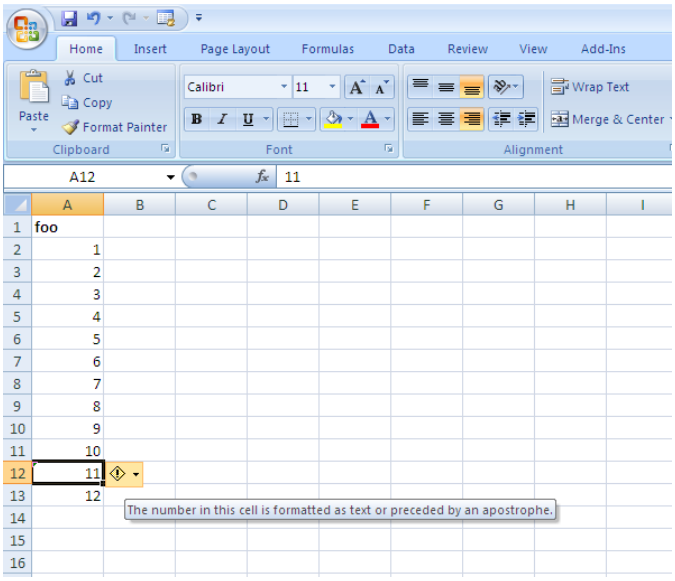
Unlike for SAS, where a variable is either character or numeric, it is possible for a number to be formatted as text in Excel even when other numbers in the column are stored as numeric values. If the numeric values in a column have been right-aligned, then a number stored as text might stand out if it is left-aligned:



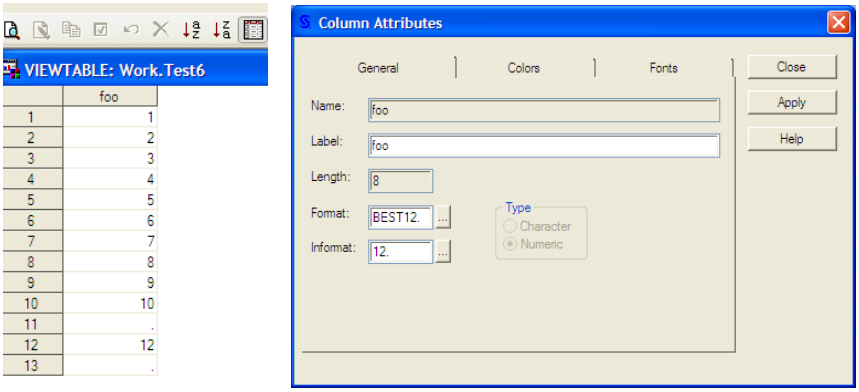
However, this can be much harder to detect by visual inspection if the text value has been right-aligned:



Only close inspection would reveal that there is a number formatted as text, but what if the data contained tens of thousands of rows?



If the above data is imported using Proc Import, SAS will create a numeric variable regardless of whether ‘mixed = no’ or ‘mixed = yes’ is specified, since the first 8 rows of the Excel data are actual numeric values. However, the number 11 has been formatted as text and SAS cannot store a character value in a numeric variable, so the dataset produced will have a missing value:



MACRO TO DIAGNOSE POTENTIAL ISSUES

The author of this paper 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.

CONCLUSION

SAS Proc Import considers the first 8 rows of data in an Excel file in order to determine whether or not corresponding SAS variables should be character or numeric. A column of data in Excel can contain a mixture of numeric and text

data. If the corresponding SAS variable is numeric, then any text values in Excel will be missing in the SAS dataset created by Proc Import, since SAS cannot store a character value in a numeric variable.

A number can be formatted as text in Excel and may appear to be the same as any other number that is actually stored as a numeric value. This can result in misleading results: if Proc Import determines that a variable is numeric, then the number stored as text in Excel will be missing in the SAS dataset.

A SAS macro has been developed to highlight potential issues by generating messages in the SAS log before calling Proc Import. The macro works by running some VBA code that saves the Excel file as a CSV file which SAS can read as text using 'input' statements. The dataset created from the CSV file is compared with the dataset created by importing the Excel data directly with Proc Import; any missing values in the latter would result in inconsistencies between the 2 datasets.

REFERENCES

	Title	Author
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