MWSUG 2016 - Paper RF04

Importing CSV Data to All Character Variables

Arthur L. Carpenter
California Occidental Consultants, Anchorage, AK

ABSTRACT

Have you ever needed to import data from a CSV file and found that some of the variables have been incorrectly assigned to be numeric? When this happens to us we may lose information and our data may be incomplete. When using PROC IMPORT on an EXCEL file we can avoid this problem by specifying the MIXED=YES option to force all the variables to be character. This option is not available when using IMPORT to read a CSV file. Increasing GUESSINGROWS can help, but what if it is not sufficient?

It is possible to force IMPORT to only create character variables. Although there is no option to do this, you can create a process that only creates character variables, and the process is easily automated so that no intervention is required on the user's part.

KEYWORDS

PROC IMPORT, CSV, Character variables, MIXED= option

INTRODUCTION

Within SAS there are a number of ways to import CSV data. The Import Wizard will build a PROC IMPORT step for you, you can write your own PROC IMPORT step, or you can write your own DATA step using the INPUT statement to control how the data are to be read.

The DATA step requires the user to have a certain knowledge of the incoming data. Variable attributes such as name, type, and length among others must be specified to create a DATA step that reads the data successfully. While the use of the DATA step is very powerful, the user is required to know something about the data itself before coding the DATA step. This necessity for prior knowledge means that this approach is less flexible when it is applied to multiple data sets or to data sets with unknown variable attributes.

The IMPORT procedure has the ability to self-determine some of the data set's variable attributes during execution. It scans the data to determine the variable names, and whether they are character or numeric, and if character their length. In order to make these determinations all or part of the data are read twice. The first scanning pass is used to determine the data characteristics (variable names and attributes), and then a second pass builds the data set itself. During the first scanning pass SAS only reads the first few rows of the incoming data, and the number of rows to be scanned is specified by the GUESSINGROWS option. When GUESSINGROWS is small, fewer resources are expended, but the possibility of incorrect determinations, such as a character field being assigned to a numeric variable, is increased.

One solution that avoids the problem of miss-assignment of character fields into numeric variables, and the subsequent loss of information is to force all variables to be character. When reading Excel files using PROC IMPORT this is easily accomplished by using the MIXED=YES option. Unfortunately that option does not exist when using IMPORT to read CSV files. The process outlined below provides the ease and flexibility of PROC IMPORT while forcing all fields into character variables.

THE APPROACH

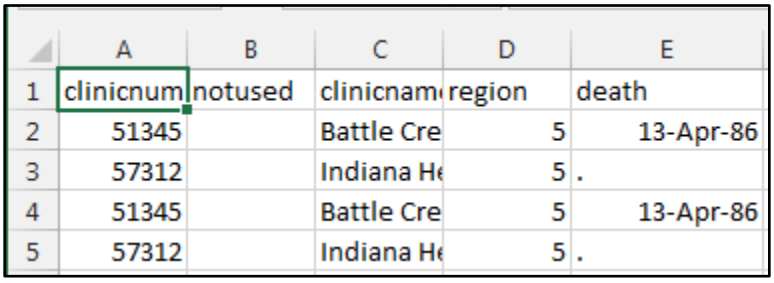
The technique outlined here is quite simple and is easily automated for any data coming into SAS from a CSV file using PROC IMPORT. Although the examples shown here assume that the CSV file has a header record containing the column

names this is generally not a necessary requirement, and minor adjustments to the code would accommodate files without a column name header record. The three basic steps discussed in this paper are:

1) Create a new observation and insert a neutral non-numeric character into each field
2) Read the data using PROC IMPORT
3) Remove the neutral character

EXAMPLE DATA

The data used in this example has five columns, with one column taking on only blank values. The other columns are a mixture of numeric and character variables. We need all columns to be read in as character. This is the view of the CSV file as rendered by Excel.



	A	B	C	D	E
1	clinicnum	notused	clinicnam	region	death
2	51345		Battle Cre	5	13-Apr-86
3	57312		Indiana He	5	.
4	51345		Battle Cre	5	13-Apr-86
5	57312		Indiana He	5	.

When the CSV file is viewed using a text editor, we can more easily visualize the actual structure of the data that we will be manipulating.

```
clinicnumber,notused,clinicname,region,death
51345,,Battle Creek Hospital,5,13-Apr-86
57312,,Indiana Help Center,5,.
51345,,Battle Creek Hospital,5,13-Apr-86
57312,,Indiana Help Center,5,.
```

INSERTING A NEUTRAL CHARACTER

The technique described in this paper works by adding a new data row and by inserting an asterisk (*) in each data column on that new data row. This is done before the file is seen by PROC IMPORT, and requires an additional pass of the data.

We read the first row which contains the names of the variables. These are counted and a series of asterisks are written on the first data line following the variable names. The new data file will now have an additional row of data; a row of comma separated asterisks is now immediately following the row of column names.

```
clinicnumber,notused,clinicname,region,death
*,*,*,*,*
51345,,Battle Creek Hospital,5,13-Apr-86
57312,,Indiana Help Center,5,.
51345,,Battle Creek Hospital,5,13-Apr-86
57312,,Indiana Help Center,5,.
```

The asterisks are inserted through the use of a DATA step, which reads the incoming CSV file and immediately rewrites it to a temporary file.

The series of inserted asterisks forces PROC IMPORT to see each column as a character variable, but because each individual asterisk has a length of 1, it cannot change or determine the variable's actual length.

```
* HOLDIT is a temporary location;
filename holdit temp  lrecl=32768; ❶

* Point to the csv file;
filename rawcsv "YOUR PATH.csv"  lrecl=32768; ❷

data _null_;
   file holdit;
   infile rawcsv end=done;

   * Read the first observation (variable names);
   input; ❸

   * Write the var names;
   put _infile_; ❹
   * Count the variables (the names are comma separated);
   wcount = countw(_infile_,','); ❺
   * Write an asterisk for each variable - forces each to be character;
   * but will not set the var length;
   string = catt('*',repeat(',*',wcount-2)); ❻
   * Write the asterisks to the new data file;
   put string; ❼

   * Read and write the data portion of the raw data;
   do until(done); ❽
      input;
      **********
      *  Pre processing of the data could be done here
      **********;
      * Write the record;
      put _infile_;
   end;
   stop;
   run;
filename rawcsv; ❾
```

❶ A temporary file will be used to hold the 'augmented' data. A large LRECL value makes sure that the file is wide enough. 32K is the widest possible length field.

❷ The incoming data is named. It does not matter that the LRECL may be oversized.

❸ The first line (with the column names) is read and stored in the automatic variable _INFILE_.

❹ The list of variable names is written into the temporary data file.

❺ The number of variables are counted. This assumes that each column is named.

❻ A series of comma separated asterisks are generated using the REPEAT function.

❼ The list of asterisks are written to the temporary data file. This will eventually become the first row of data values. The asterisks will force PROC IMPORT to assume that each column is character.

❽ A DO loop is used to read and write each of the remaining data lines to the temporary data file. The value of the temporary variable DONE is set to one when the last record is read from the incoming data file. DONE is created by the INFILE statement.

❾ The *libref* pointing to the raw data is cleared.

The temporary data file can now be read using PROC IMPORT.

READING THE DATA

PROC IMPORT is then used to import the modified CSV file into a SAS data set. Although the GUESSINGROWS option is no longer needed to determine whether a variable is numeric or character, it is still used to determine the maximum length for each variable. The DATAROW=2 option is used to make sure that the row of asterisks is included in the read, but not the variable names.

```
* Read the Altered CSV file into a SAS dataset;
PROC IMPORT OUT= fromcsv ❶
            DATAFILE= holdit ❷
            DBMS=CSV
            REPLACE;
   guessingrows=444444; ❸
   GETNAMES=YES; ❹
   DATAROW=2; ❺
   run;
* Clear the temporary fileref;
filename holdit; ❻
```

❶ The new outgoing data set is named.

❷ The temporary file, with the asterisks, is to be used by IMPORT to form the new data set.

❸ GUESSINGROWS is set to an arbitrarily large number.

❹ The column names in the first row are to be used to form the variable names.

❺ The data starts in row 2. This is the row that contains the asterisks.

❻ The *libref* associated with the temporary file is cleared.

REMOVING THE NEUTRAL CHARACTER

The first time this new data set (WORK.FROMCSV) is used, the asterisks can be removed by using the FIRSTOBS data set option. Because the FIRSTOBS option specifies the first row of data that is to be read into the DATA step, it is easy to eliminate the asterisks which will always be in observation 1. The DATA step is then available to provide whatever other processing is to be done with this data set.

```
data noasterisk;
   set fromcsv(firstobs=2);
   * Doing other stuff here.;
   run:
```

FIRSTOBS can be used as either a data set option or as a system option. This means that the step that eliminates the observation containing the asterisks could be a procedure step as well as a DATA step.

SUMMARY

Although the MIXED=YES option is not available to the IMPORT procedure when reading CSV files, you can still force all variables to be character. By inserting a series of asterisks into the CSV file, the IMPORT procedure automatically assigns the variable type of character to each variable.

ABOUT THE AUTHOR

Art Carpenter's publications list includes; five books, two chapters in *Reporting from the Field*, and numerous papers and posters presented at SAS Global Forum, SUGI, PharmaSUG, WUSS, and other regional conferences. Art has been using SAS since 1977 and has served in various leadership positions in local, regional, and national user groups.

Art is a SAS Certified Advanced Professional Programmer, and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

Recent publications are listed on my sasCommunity.org Presentation Index page.
http://sascommunity.org/wiki/Presentations:ArtCarpenter_Papers_and_Presentations

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

REFERENCES

The technique shown in this paper was demonstrated in a sasCommunity.org article and was highlighted on a sasCommunity.org "Tip of the Day".

TRADEMARK INFORMATION