PharmaSUG China 2016 - Paper 41

A Macro to Import Data from CSV File to SAS

Liang Guo, FMD, Wuhan, Hubei Province, China
Pei Zhang, FMD, Wuhan, Hubei Province, China

ABSTRACT

CSV is a universal and relatively simple file format, and widely used in data transfer. Generally, CSV file can be imported into SAS by using PROC IMPORT very easily. However, when CSV file is converted into SAS by using PROC IMPORT, part of the data could be disordered or the program could be aborted due to error when line break exists in data. This paper discusses a macro which can be used to import csv data into SAS, and it can overcome the line break problem yet keep most capacity of PROC IMPORT.

INTRODUCTION

In computing, a comma-separated values (CSV) file stores tabular data, numbers or text, in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

CSV is a common data exchange format that is widely supported by consumer, business, and scientific applications. Among its most common uses is moving tabular data between programs that natively operate on incompatible (often proprietary and/or undocumented) formats.

Generally, CSV file can be imported into SAS by using PROC IMPORT and EXPORT/IMPORT facility very easily. However, when CSV file is converted into SAS by using PROC IMPORT, part of the data could be disordered or the program could be aborted due to error if line break exists in any field. Because PROC IMPORT considers a record is ended at a line terminator (e.g. line break). In order to read the CSV file correctly in this condition, we created the macro CSV.

MACRO DESCRIPTION

The goal of the macro is to provide a solution to overcome the line break problem when import csv data into SAS, yet keep most capacity of PROC IMPORT. This macro is an optimized version based on macro CSV (by Victor Kamensky) which was posted on NESUG 2003. The macro CSV writes a program to read the CSV into SAS (by running the program).

One of the basic requirement of CSV is “fields containing a line-break, double-quote, and/or commas should be quoted”. The macro identifies the line-break and read the data based on this requirement as below (detail SAS code please see step D in section FULL MACRO CODE):

(1) Counting the number of double quotation mark (“) in a field which read by SAS. If the number of double quotation mark is not even and the first character is (“) in a field, it means line break exists in the origin field, and SAS considers a record is ended at line break.

(2) Continue to read the following field which read by SAS and collect by a temporary variable, and concatenate with the origin field by “'0A'x” or other customer specified connector, till the number of double quotation mark in the whole field becomes even. Counting the number of double quotation mark in the combined variable.

(3) Repeat step (2) till the number of double quotation mark is even.

Besides overcoming the line break problem, it is faster to use this macro than to run PROC IMPORT.

MACRO PARAMETERS

There are 2 positional parameters and 10 keyword parameters. Positional parameters are required. Keyword parameters are optional.

Name	Description	Default value
	POSITIONAL PARAMETERS	
CSV	CSV file name. The full path should be specified if the CSV file is not in the user's current directory (i.e. directory path seen on the bottom of the SAS session).	
DATASET	SAS data set name that macro creates. Can be permanent or temporary.	
	KEYWORD PARAMETERS	
PROGNAME	Name of the program to read the CSV file. The program is written by the macro.	_a, i.e. default program name is _a.SAS.
LSIZE	Maximum length of a record in the CSV file.	32767 (maximum possible)
MAXLEN	Maximum length of any cell.	200
DROPMISS	Blank columns are dropped from the SAS data set if DROPMISS is equal to Y.	Y
LENDATE	EXCEL columns are checked to be DATE formatted as MMDDYY&LENDATE.	10
LENADD	Constant added to maximum existing length of a CHARACTER EXCEL column, to define SAS character variable length.	0
NAMES	Name of SAS dataset to keep variable names.	_names_
GETNAME	The value of first column of CSV will be used as variable name of SAS dataset if GETNAME is equal to Y.	Y
LINE_BREAK	This specifies whether line break is exist in any cell or not.	N
CONNECTOR	CONNECTOR which is used to concatenate the text separated by line break.	'0A'x

EXAMPLES OF MACRO IN USE

Example I. Default values are used for all positional parameters.

%csv(MYCSV, MYSASDS);

1	CSV file MYCSV.CSV should by default be in the user's current directory (directory whose name can be seen on the bottom of the SAS session).
2	MYSASDS data set is created.
3	Program _a.SAS is written into the user's current directory.
4	Length of records in the CSV file should not be greater than 32767. The length of any cell cannot be more than 200.
5	Missing columns are dropped from the final data set.

6	Dates formatted as MMDDYY10 (if any) are converted into numeric variables formatted as mmddyy10.
7	Length of each character variable is equal to maximum length of its corresponding column.
8	The value of first column of CSV is used as variable name of SAS dataset

Example II. Positional parameters GETNAME is added.

%csv(MYCSV, MYSASDS, GETNAME = N);

The same as example I except for

9	The variable names are assigned according to column number, like F1, F2 … Fn.

Example III. Positional parameters LINE_BREAK is added if line break is exist in any cell, and CONNECTOR is added.

%csv(MYCSV, MYSASDS, GETNAME = N, LINE_BREAK = Y, CONNECTOR=”; “);

10	Specify line break exist in some fields.
11	Use “; ” to concatenate the text separated by line break.

FULL MACRO CODE

```
%macro csv(CSV, DATASET, PROGNAME=_a, LSIZE=32767, MAXLEN=200, DROPMISS =Y,
LENDATE=10, LENADD=0, _NAMES_=_names_, GETNAME=Y, LINE_BREAK=N, CONNECTOR ='0A'x);

options mprint validvarname=v7;

/*A: The first line of the &CSV file is read into one character variable LINE in
data set &_NAMES_.
     LINE is checked for the presence of double quotation marks (").
     Commas (,) inside double quotation marks are changed into underscores (_).
     Data set &_NAMES_ has 1 observation.*/
data &_NAMES_;
    length line $&LSIZE. ;
    infile "&CSV..csv" ls=&LSIZE lrecl=&LSIZE missover length=l;
    input line $varying&LSIZE.. l;
    if _n_=1;
    if index(line,'"') > 0 then do;
        _in=0;
        DROP _in _ii;
        do _ii=1 to length(trim(line));
            if _in=1 and substr(line,_ii,1)=',' then substr(line,_ii,1)='_';
            if substr(line,_ii,1)='"' then _in=1-_in;
        end;
    end;
run;
```

```
/*B: The variable LINE is divided into NAMEs separated by commas (,).
     The following is done with each NAME. NAME is truncated to 32 characters.
     All characters except letters and digits are substituted into underscores (_).
     If NAME starts with a digit or underscore (_), letter F is added (or
substituted) as a first character.
     Blank names are substituted into F#, where # means variable number (VARNUM).
     The number of observations in data set &_NAMES_ is now equal to the number of
EXCEL columns (&NUMCOL).
     If &GETNAME is assigned to N, then then NAME are assigned to F#, where # means
variable number (VARNUM).*/
data &_NAMES_(keep=name varnum);
    length name $ 32;
    set &_NAMES_;
    varnum=0;
    ind=0;
    do until(index(line,',')=0);
        varnum=varnum+1;
        line=substr(line,ind+1);
        name=line;
        ind=index(line,',');
        if ind > 1 then name=scan(line,1,',');
        if ind = 1 then name='F'||compress(put(varnum,best.));
        name=left(trim(name));
        do _ii=1 to length(trim(name));
            if index ('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_',
            upcase(substr(name,_ii,1)) ) =0 then substr(name,_ii,1)='_';
        end;
        if index ('0123456789',upcase(substr(name,1,1)) ) > 0 then
    name='F'||name;
        name=upcase(substr(name,1,1))||substr(name,2);
        if substr(name,1,1)='_' then substr(name,1,1)='F';
        output;
    end;
    call symput('numcol',compress(put(varnum,best.)));
run;

%if &getname.=N %then %do;
data &_NAMES_;
    set &_NAMES_;
    name='F'||compress(put(varnum,best.));
run;
%end;

%let max_col=%eval(&numcol. + 10);
```

```
/*C: The purpose of this macro loop is to get rid of double NAMEs. In case of
double names the variable number is added at the end of the name.*/
%let dif_names=0;
%do %until (&dif_names gt 0);
proc sort data=&_NAMES_;
    by name varnum;
run;

data &_NAMES_;
    set &_NAMES_ nobs=nobs end=last;
    by name;
    length defname $8;
    retain n_dif;
    drop n_dif;
    if first.name then n_dif+1;
    if not first.name then do;
        length _vn $ 10; drop _vn ln;
        _vn =compress(put(varnum,best.));
        ln =32-length(compress(_vn));
        name=compress(substr(name,1,ln)||compress(_vn));
    end;
    if last and nobs=n_dif then call symput('dif_names','1');
    defname="V"||strip(put(varnum, best.));
run;
%end;

proc sort data=&_NAMES_;
    by varnum;
run;

%if &GETNAME.=Y %then %do;
%let firstobs=2;
%end;
%if &GETNAME.=N %then %do;
%let firstobs=1;
%end;

/*D: The CSV file is read to the &DATASET. But it is not the final &DATASET.
     Each variable in the &DATASET is now character, length =&MAXLEN (default 200).
     Variable names are V1- V&NUMCOL.*/
%if &LINE_BREAK.=Y %then %do;
data &DATASET.(drop=_:);
    infile "&CSV..csv" ls=&LSIZE lrecl=&LSIZE missover dlm=',' dsd
```

```
firstobs=&firstobs.;
    informat _v1-_v&max_col. $&MAXLEN..;
    format v1-v&numcol. _v1-_v&max_col. $&MAXLEN..;
    input _v1-_v&max_col. @;
    array var[&max_col.] _v1-_v&max_col.;
/*  clean line break in any cell*/
    %do i=1 %to &numcol.;
/*  identify whether line break exist in a data field*/
        if mod(countc(_v&i., '"'), 2)=0 or (length(_v&i.)>1 and substr(_v&i.,1,1)
ne '"')
            then v&i.=_v&i.;
        else do;
            _v&i._count=0;
            do until(mod(_v&i._count, 2) = 0);
                _v&i._count=_v&i._count + countc(_v&i., '"');
                v&i.=catx(&CONNECTOR., v&i., _v&i.);
                if mod(_v&i._count, 2) then do j= %eval(&i.+1) to &max_col.;
                    v&i.=catx(', ', v&i., var[j]);

                    _v&i._count=_v&i._count + countc(var[j], '"');
                    if mod(_v&i._count, 2)=0 then do;
                        %do k = 1 %to %eval(&max_col.-&i.);
                           if j+&k.<=&max_col. then var[&i.+&k.]=var[j+&k.];
                        %end;
                        j=&max_col.;
                    end;
                end;
                if mod(_v&i._count, 2) then do;
                    input _v&i. - _v&max_col.;
                end;
            end;
            v&i.=strip(dequote(v&i.));
        end;
    %end;
    drop j;
run;
%end;
%else %if &LINE_BREAK.=N %then %do;
data &DATASET.;
    infile "&CSV..csv" ls=&LSIZE lrecl=&LSIZE missover dlm=',' dsd
firstobs=&firstobs.;
    informat v1-v&numcol. $&MAXLEN..;
    format v1-v&numcol. $&MAXLEN..;
    input v1-v&numcol. @;
```

```
run;
%end;

data _null_;
    call symput('numrow',compress(put(numrow,best.)));
    stop;
    set &DATASET nobs=numrow;
run;

/*E: &DATASET is transposed (into itself, by PROC TRANSPOSE). Transposed &DATASET
is merged with &_NAMES_.*/
proc transpose data=&DATASET out=_&DATASET name=name;
    var _all_;
run;

data &_NAMES_;
    merge &_NAMES_ _&DATASET(drop=name);
run;

/*F: Column types are defined for all columns of the EXCEL spreadsheet. The columns
now correspond to variables COL1-COL&NUMROW in &_NAMES_ dataset.
     COL is NUMERIC (type=1) if all cells contain only numbers or blanks.
     COL is DATE (date=1) if all cells are dates formatted as MMDDYY&LENDATE.
     COL is CHARACTER if it is not NUMERIC or DATE.*/
data &_NAMES_;
    length name $ 32;
    set &_NAMES_;
    array col col1-col&numrow;
    drop col1-col&numrow;
    type=1;
    date=1;
    do over col;
        length=max(length,length(trim(col)));
        if compress(col)=:'-' and compress(col)^='-' then col=substr(left(col),2);
        if compress(col,'0123456789.')^='' then type=2;
        if compress(col,'0123456789.') ='' and compress(col,'. ')^='' then do;
            if count(col,'.')>1 then type=2;
            else n_number=sum(n_number,1);
        end;
        if compress(col)^='' and date=1 then do;
            date=.;
            if compress(col,'0123456789/')='' and length(trim(col)) in
(&LENDATE,%eval(&LENDATE-1),%eval(&LENDATE-2)) then do;
                if input(compress(col),mmddyy&LENDATE..) > . then date=1;
```

```
            end;
        end;
        if compress(col)=' ' then n_blank=sum(n_blank,1);
        if compress(col)='.' then n_dots =sum(n_dots,1);
    end;
    nobs=&numrow;
    if date=1 then type=1;
    if type =2 and n_blank =nobs then do;
        dropflag=1;
        type=1;
    end;
    if type =1 and (n_blank > . or n_dots > .) then if sum (n_blank,n_dots)=nobs
then do;
        dropflag=1;
        date=.;
    end;
    drop n_blank n_dots;
run;

/*G: A program to read the &CSV file is written. Its name is &PROGNAME..sas.*/
data _null_;
    file "&PROGNAME..sas";
    put "title '" "&PROGNAME..sas" "';";
    put "data &DATASET;";
    put " set &DATASET;";
run;

data _null_;
    set &_NAMES_(where=(type=2)) end=last;
    file "&PROGNAME..sas" mod;
    length=length+&LENADD;
    if _n_=1 then put ' LENGTH ';
    put @ 5 name ' $ ' length;
    if last then put ';';
run;

data _null_;
    set &_NAMES_(where=(date=1)) end=last;
    file "&PROGNAME..sas" mod;
    put @ 5 ' informat ' name " mmddyy&LENDATE..;" @;
    put ' format ' name " mmddyy&LENDATE..;" ;
run;

data _null_;
```

```
    set &_NAMES_ end=last;
    file "&PROGNAME..sas" mod;
    if type=1 then do;
        if date=1 then put @ 8 name  ' = input(' defname ", mmddyy&LENDATE..);";
        else put @ 8 name  ' = input(' defname ", best12.);";
    end;
    else if type=2 then put @ 8 name  ' = ' defname ';';
    if last then put 'run;';
run;

/*H: Data steps to drop blank columns from the &DATASET are written. Executed only
if &DROPMISS=Y.*/
%if &DROPMISS eq Y %then %do;
data _null_;
    set &_NAMES_(where=(dropflag=1)) end=last;
    file "&PROGNAME..sas" mod;
    if _n_=1 then do;
        put "data &DATASET; set &DATASET;";
        put @ 5 "DROP ";
    end;
    put @ 7 name ;
    if last then put ';run;';
run;

data &_NAMES_;
    set &_NAMES_ ;
    if dropflag ne 1;
run;
%end;

/*I: RETAIN statement to keep the variables in &DATASET in the same order as the
columns in the CSV file (EXCEL spreadsheet) is written.*/
data _null_;
    set &_NAMES_ end=last;
    file "&PROGNAME..sas" mod;
    if _n_=1 then do;
        put "data &DATASET.(drop=v1-v&numcol.);" ;
        put @ 5 "RETAIN ";
    end;
    put @ 7 name ;
    if last then do;
        put @8 ';';
        put @3 "set &DATASET;";
        put 'run;';
```

```
    end;
run;

/*J: &CSV is converted to &DATASET by running &PROGNAME.. sas.*/
options mprint source2;
%include &PROGNAME;
%mend csv;
```

CONCLUSION

In this paper, we described some characteristics of CSV files, and discussed a special case which can't be handled by PROC IMPORT when import CSV file to SAS. A macro has been presented to overcome this problem. The macro is easy to use, and provide many parameters to user for different purpose. In addition to the superiority in reading line break, it is also faster to use macro CSV than to run PROC IMPORT.

REFERENCES

Victor Kamensky. 2003. CSV: A MACRO WHICH WRITES SAS® PROGRAMS TO READ CSV FILES. Proceedings of the NESUG 2003 Conference. Cary, North Carolina: SAS Institute Inc. Available at http://www.lexjansen.com/nesug/nesug03/ps/ps019.pdf.

ACKNOWLEDGMENTS

The author would like to thank his friend and colleague Chao Wang for the review of the paper and all support during preparing this paper.

CONTACT INFORMATION

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

Enterprise: Fountain Medical Development Ltd.

E-mail: liang.guo@fountain-med.com