PharmaSUG 2023 – Paper QT-001

A SAS® Macro to Convert CSV files to SAS Datasets

Zemin Zeng and Bo Yuan
Sanofi, Bridgewater, NJ

ABSTRACT

Statistical programmer in the pharmaceutical industry often needs to convert Comma Separated Values (CSV) files to SAS® datasets. This paper provides a short SAS macro to automatically convert CSV files to SAS datasets. We shall share the techniques in the macro on how programmatically to identify the column variables in a CSV file without any manual inputs. The macro has been widely used at work and has been proven to increase programming efficiencies and provide time/cost savings.

INTRODUCTION

In the pharmaceutical industry, data management or statistical programming teams often receive a bunch of CSV files from external data transfer and need to convert them into SAS datasets for downstream statistical analysis. There are a few ways which can convert CVS file to SAS datasets: "PROC IMPORT procedure, the SAS DATA Step in combination with the INFILE statement, and the Import Wizard" (https://sasexamplecode.com/3-ways-to-import-a-csv-file-into-sas-examples/), but there are some short comings for them. For instance, when using PROC IMPORT to convert CSV files, certain columns such as SUBJID, SITEID and character date variables will be forcefully converted to numeric variable. If data steps with INFILE statement used to import CSV files, it requires lots of manual work to specify column variables for each CSV file. The Import Wizard is an easy method to import CSV files, but it isn't flexible and automated, if we have a lot of CSV files it is hard to use Import Wizard to import CSV file one by one.

This macro was developed to help programmers better convert CSV data files to SAS datasets automatically. The macro is target to convert all CSV files under a given directory, we scan a large enough amount of columns of each CSV file using INFILE statement, programmatically identify the number of columns and column variable name in each CSV file, then resize variable lengths to their actual size. The complete code of the SAS macro was included as an appendix, which can be modified to fit reader's need.

SETTING

The macro was named %csv2sas. There are only two parameters need to be assigned, specifically: the parameter for CSV file directory(&csvdir) and the parameter for output SAS datasets location(&dataout). Users can put the program into any folder. To make program running faster and sufficient for all CSV automatically converting, we assume all CSV files should have less than 300 columns and have a maximum text of each column less than 2000 characters. Users can easily modify the maximum number of columns and text length to better meet their needs.

KEY STEPS

This macro includes five steps. First, get all the CSV files names from given folder and prepare for a loop. Second, read in CSV file to SAS dataset with all the variables having the same length, then divide the dataset into two, with one dataset has only the first line for SAS dataset (i.e. variable name line), and the other dataset contains data only. Third, identify the number of columns with data in a CSV file. Fourth, identify variables and their actual lengths. The last step is to output SAS datasets. Below are the details of each steps in the macro:

Step 1: Use SAS DREAD function to return the file names of a given directory. Then we select all CSV file names and combine them into a macro variable for a loop preparation. A macro variable *cname* will be generated to help create a list of CSV file names. Another macro variable *num* will also be generated to capture total number of CSV files, which will be used instep 2 to loop through all CSV files.

Figure 1: Codes used to get all the names of CSV files using FILENAME function.

```
data filename;
  length fref $8 fname $200;
  did=filename(fref, "&csvdir.");
  did=dopen(fref);
    do i=1 to dnum(did);
        fname=dread(did, i); output;
      end;
  did=dclose(did); did=filename(fref);
run;

data filename2;
  set filename; where index(fname, ".csv")>0;
  cname=scan(fname,1,'.'); num=_n_;
  keep fname num i cname ;
run;

proc sql noprint;
  select strip(put(max(num),best.)) into : num from filename2;
  select cname into :filenm1 - :filenm&num. from filename2;
quit;
```

Step 2: Use FILENAME to assign file reference *dsn* for an individual CSV file in the loop. ARRAY in a data step to set read in 300 columns with each column 2000 characters to get a _TEMP dataset for future use. Users can increase the number of columns and column widths based on their needs. The _TEMP dataset will include all the information of CSV files (**Figure 2).**

Figure 2: Codes of data step input CSV file to SAS dataset with same length for all column

```
%do j=1 %to &num.;  /*loop over CSV files*/
filename dsn "&csvdir./&&filenm&j...csv" lrecl=32767;

data _temp;
  array column (300) $ 2000 col1-col300;
  infile dsn truncover dsd dlm='2c0d'x;
  input %do i=1 %to 300; column (&i) %end; ;
run;
```

After that, we divide the _TEMP dataset into two sub-datasets: _TEMP1 and _TEMP2. Sub-dataset _TEMP1 include variable names only, and _TEMP2 has all other data up-to 2000 characters in each column to avoid truncation.

Figure 3: separate _TEMP into two datasets

```
data _temp1 _temp2;
  set _temp;
  if _n_=1 then output _temp1; else output _temp2;
run;
```

Step 3: Identify the actual number of columns of each CSV file, and then keep only variables in _TEMP1 and _TEMP2 that have data in the CSV file(**Figure 4)**. The rule we use to evaluate if having data is to check variable names in TEMP1 if they are not missing.

Figure 4: Code to identify the actual number of columns.

```
proc transpose data=_temp1 out=_cnt1;
  var col:;
run;

*to decide the number of columns in the CSV file;
data _cnt2;
  set _cnt1;
  if col1 ne '' then  CALL SYMPUT ("_numvar", _n_);
  if col1 ne '';
run;
  %put columns = : &_numvar;

data _temp1; set _temp1;
  %do i=1 %to &_numvar; keep col&i.; %end;
run;

data _temp2; set _temp2;
  %do i=1 %to &_numvar; keep col&i.; %end;
run;
```

Step 4: Identify variables and their actual length. Resize all variables to maximum length of each column. First, use PROC CONTENTS to get all the names of variables from _TEMP2, and then use data step to create a new variable includes MAX and LENGTH functions for each observation. A macro variable is generated to calculate the maximum length from _TEMP2 and keep the maximum length for each observation in _LEN. Set _TEMP1 and _LEN to create lengths for each variable (**Figure 5).**

Figure 5: Code to get maximum length from _TEMP2 for every column

```
proc contents data=_temp2 out=_var(keep=name) noprint; run;

data _var1;
  set  var;
  col="strip(put(max(length(strip(" || strip(name) ||
    "))), best.)) as " || strip(name);
run;

proc sql noprint;
  select col into: list_var separated by ',  ' from _var1;
  create table _len as select distinct &list_var from _temp2;
quit;

data _temp_len;
  set _temp1 _len;
run;
```

Using _TEMP_LEN dataset to create the macro attribute variable _LENGTH to keep the length and order information. The values of this macro variable look like: STUDYID $8 DOMAIN $2 USUBJID $21…. A _KEEP macro variable should be created to keep all the columns of CSV (**Figure 6**).

Figure 6: Get the variable names and their actual lengths

```
proc transpose data=_temp_len out=t_temp1;
%do i=1 %to &_numvar; var col&i.; %end;
run;

data t_temp1;
  set t_temp1;
  ord=input(substr(strip(_NAME_),4), best.);
run;

proc sql noprint;
select strip(col1)||" $"||strip(col2) into: _length separated by ' '
  from t_temp1 order by ord;
  select strip(col1) into: _keep separated by ' '
  from t_temp1 ;
quit;
%put Variables in %upcase(&&&filenm&j.) dataset: &_keep;
%put and theri length: &_length;
```

Step 5: Output SAS dataset. We use variable values in TEMP1 to name each column. The final SAS dataset is output to the location specified in the macro parameter *dataout* (**Figure 7**).

Figure 7: Output SAS dataset

```
data dataout.&&filenm&j.(keep=&_keep.);
  length &_length;
  set  _temp2;
  %do i=1 %to &_numvar; &&_var&i=strip(col&i); %end;
run;
```

CONCLUSION

The SAS macro **%csv2sas** can be used to convert all CSV files in a given directory into SAS datasets. It is easy to use and powerful to deal with CSV files with different columns and variable lengths without any manual work in the conversion process. With only 5 steps, the macro is very simple, but dramatically improves programming efficiencies. It also generates an output that helps the programming team do further data analysis and check for potential data issues. As the macro only presents what we see in the CSV files into SAS dataset format with all variables in Character version, some follow-up programming activities maybe needed to convert some variables into numeric.

REFERENCES

SAS Example Code. 2021. “3 Ways to Import a CSV File into SAS.” https://sasexamplecode.com/3-ways-to-import-a-csv-file-into-sas-examples/

CONTACT INFORMATION

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

Zemin Zeng, Team Lead for I&I and Neuro, Statistical Programming, Sanofi, Bridgewater, NJ

Email: Zemin.Zeng@sanofi.com

Bo Yuan, Statistical programmer contractor, Sanofi, Bridgewater, NJ
Email: Bo.Yuan@sanofi.com

TRADEMARK INFORMATION

APPENDIX: THE CODE OF THE MACRO

```
/*************************************************************************
* Program Name               : csv2sas.sas
* Program Purpose            : Convert CSV files to SAS datasets
* Program Author             : Zemin Zeng/Bo Yuan
* Program creation date      : 2021-12-15
* Input data files           : CSV files
* Output data files          : SAS Data Set(.sas7bdat)
* System                     : SAS version 9.4 - WISE environment
* Macro call Sample          : %csv2sas(csvdir=, dataout=)
*************************************************************************/

/*======================== Beginning of Code ======================*/
%macro csv2sas(csvdir=, dataout=);
libname dataout "&dataout.";
*=======================================================================;
*         Step 1: get all the CSV files form given folder;
*=======================================================================;
data filename;
  length fref $8 fname $200;
  did=filename(fref, "&csvdir.");
  did=dopen(fref);
    do i=1 to dnum(did);
        fname=dread(did, i); output;
      end;
  did=dclose(did); did=filename(fref);
run;

data filename2;
  set filename; where index(fname, ".csv")>0;
  cname=scan(fname,1,'.'); num=_n_;
  keep fname num i cname ;
run;

proc sql noprint;
  select strip(put(max(num),best.)) into : num from filename2;
  select cname into :filenm1 - :filenm&num. from filename2;
quit;

*=======================================================================;
*         Step 2: read in csv file and output sas dataset;
*=======================================================================;
%do j=1 %to &num.;  /*loop over CSV files*/
   filename dsn "&csvdir./&&filenm&j...csv" lrecl=32767;

   data _temp;
     array column (300) $ 2000 col1-col300;
```

```
    infile dsn truncover dsd dlm='2c0d'x;
    input %do i=1 %to 300; column (&i) %end; ;
  run;

  data _temp1 _temp2;
    set _temp;
    if _n_=1 then output _temp1; else output _temp2;
  run;

*=====================================================================;
*         Step 3: identify the number of columns;
*=====================================================================;
    proc transpose data=_temp1 out=_cnt1;
      var col:;
    run;

    *to decide the number of columns in the CSV file;
    data _cnt2;
      set _cnt1;
      if col1 ne '' then  CALL SYMPUT ("_numvar", _n_);
      if col1 ne '';
    run;
      %put columns = : &_numvar;

    data _temp1; set _temp1;
      %do i=1 %to &_numvar; keep col&i.; %end;
    run;

    data _temp2; set _temp2;
      %do i=1 %to &_numvar; keep col&i.; %end;
    run;

*=====================================================================;
*         Step 4: identify variables and their actual lengths;
*=====================================================================;
      data _null_;
        set _temp1;
        %do i=1 %to &_numvar;
           CALL SYMPUT ("_var" || compress(put(&i, best.)), col&i);
        %end;
      run;

      proc contents data=_temp2 out=_var(keep=name) noprint; run;

      data _var1;
        set _var;
        col="strip(put(max(length(strip(" || strip(name) ||
          "))), best.)) as " || strip(name);
      run;

      proc sql noprint;
        select col into: list_var separated by ',  ' from _var1;
        create table _len as select distinct &list_var from _temp2;
      quit;

      data _temp_len;
        set _temp1 _len;
      run;

      proc transpose data=_temp_len out=t_temp1;
      %do i=1 %to &_numvar; var col&i.; %end;
```

```
        run;

        data t_temp1;
          set t_temp1;
          ord=input(substr(strip(_NAME_),4), best.);
        run;

        proc sql noprint;
        select strip(col1)||" $"||strip(col2) into: _length separated by ' '
          from t_temp1 order by ord;
          select strip(col1) into: _keep separated by ' '
          from t_temp1 ;
        quit;
       %put Variables in %upcase(&&&filenm&j.) dataset: &_keep;
       %put and theri length: &_length;

*=================================================================;
*         Step 5: output SAS dataset
*=================================================================;
        data dataout.&&filenm&j.(keep=&_keep.);
          length &_length;
          set  _temp2;
          %do i=1 %to &_numvar; &&_var&i=strip(col&i); %end;
        run;
%end; /*loop over CSV files--end*/
%mend csv2sas;

*Macro call;
%csv2sas(csvdir=%sysfunc(pathname(EXTERI)),
         dataout=%sysfunc(pathname(EXTERI))/dataout);
```