Paper 077-29

Subset Raw Data and Zip Them Up

Desheng(Ben) Xu, CGS, Northern Illinois University, De Kalb, IL

ABSTRACT

The special variable _INFILE_ can be used to subset the raw input data. The PUT _INFILE_ statement writes the entire input record to the output destination. The %DO loop in macro is used to create dozens of external file references. Finally, use data step statements to generate a batch program that can be invoked in WinZip command line to zip up every resulting subset file. This paper is appropriate for programmers with intermediate skills and related macro background.

INTRODUCTION

A national WIASRD (Workforce Investment Act Title I-B Standardized Record Data) file in CSV format (about 700MB) is subset based on the state name variable. The resulting subset files need to be zipped so that the clients from different geographical areas can quickly download them. Before this task, the national SAS data set called NationalPerformance.sas7bdat has already been created from the raw input data for other purposes.

The process requires WinZip Command Line Support Add-On. Instead of using WinZip graphical user interface, it allows the use of WinZip directly from the command prompt or batch file. It provides an ideal way for quick zipping of repetitive tasks. It is available for WinZip 8.0 or later version. For more information, check the web site at www.winzip.com/wzcline.htm.

The work is done with SAS 8.2 on Windows 2000.

STEP 1

Since the national SAS data set is already there, it can be used to retrieve different state names. Eventually, the state name will be used as the file name for each state CSV file and zip file. In the SAS data set, variable v914 contains the state name.

```
libname wia 'c:\projects\wiasrd';
proc sql;
  create table statelist as
  select distinct v914 from wia.NationalPerformance;
quit;
```

STEP 2

In order to separate the input data for each state, we have to create every external file reference for each state. Then use the FILE statement in data step to output into the corresponding destination. Obviously, we need to find out how many states are really there. It can be obtained through the data step or the automatic macro variable - &SQLOBS from step 1.

```
data _null_;
  set statelist NOBS=N;
  call symput('N', put(N,2.));
  stop;
run;
%put Number of States = &N;
```

After the number of states is known, we create a macro variable for each state, for instance, state1 contains "Alabama", state2 contains "Alaska" and so on. The real macro variable values do not have the double quotes.

```
%macro CreateOutFile;
 data _null_;
   do i=1 to &N;
     set statelist;
     call symput('state'||trim(left(put(i,2.))), trim(v914));
   end;
 run;
 %do i=1 %to &N;
   %let out=%sysfunc(compress(&&state&i));
   %global file&i;
   %let file&i="c:\projects\wiasrd\&out..csv";
 %end;
%mend CreateOutFile;

%CreateOutFile
```

Here, we use compress( ) function to remove the blanks in the state name, such as "District of Columbia" because we don't want the spaces to cause any potential trouble in the later steps. After executing this macro, macro variable file1 points to "c:\projects\wiasrd\Alabama.csv", etc. In order to verify these macro variables have been successfully generated, you can use %PUT _USER_ statement to get the list of all user defined macro variables.

STEP 3

Now let's get down to the business. When we output every state data to comma separated file format, FILE statement is used. However, if we attempt to use direct data step statement, we have to write more than 50 similar FILE statements like this:

```
if v914 eq "&state1" then
  file &file1 dsd dlm=',' lrecl=12000;
else if v914 eq "&state2" then
  file &file2 dsd dlm=',' lrecl=12000;
else if v914 eq "&state3" then
  file &file2 dsd dlm=',' lrecl=12000;
...
```

A better approach is to turn to a macro solution. For the first state, we want the IF statement; for the rest of states, ELSE IF statements should be produced. Be careful that each of these two statements must be included in %DO and %END block.

```
%macro repeatif;
 %do i=1 %to &N;
   %if &i=1 %then
     %do;
       if v914 eq "&state1" then file &file1 dlm=',' lrecl=12000;
     %end;
   %else
     %do;
       else if v914 eq "&&state&i" then file &&file&i dlm=',' lrecl=12000;
     %end;
 %end;
%mend repeatif;
```

In the original CSV input file, the first line lists all the variable names. We use FIRSTOBS= option to skip this line. Since the input file is comma separated, there is no shortcut to jump into the state name variable; we have to read all the variables until hit the right one. We use @ to hold current input line and decide which external file reference should be used.

The _INFILE_ is an automatic character variable that references the contents of the current input buffer for this INFILE statement. The PUT _INFILE_ statement will write the entire current record to that external file.

Through the system option SYMBOLGEN, you can easily tell how the macro variables are resolved. With MPRINT option, it is very clear how the macro %repeatif functions.

```
*options mprint symbolgen;
filename indata 'c:\projects\wiasrd\Wiasrd2001.csv';

data _null_;
  length v101 v338 v603 v313A v313B v313C $ 9. v301 $ 5. v914 $ 20. ;
  informat v102 v125 v302 v303 v332 v333 v626 v628 v630 v632 v634 v636 v638
           v640 v642 v644 v646 v648 v650 v652 v654 v656 v658 v660 v662 v664
           v666 v668 v670 v672 v674  mmddyy8.;
  infile indata firstobs=2 lrecl=12000 dsd dlm=',' missover;
  input v101 $ v102-v131 v301 v302-v312 v313a v313b v313c v314-v345 v601-v676
        v901-v913 v914 $ @;

  %repeatif;
  put _infile_;
run;
```

We would point out that an alternative way to subset the data is to create the SAS data set for each state, and then use PROC EXPORT to output to CSV files. We don't show any details here because we want to demonstrate the usage of special variable _INFILE_.

STEP 4

All the state files are ready to be zipped. Obviously the task will be quite tedious if we zip them one by one. So we wonder if we could write a batch program that could handle the job. The WinZip Command Line Add-On exactly fits in this case. It allows us to run a batch file without the graphic interface.

```
%macro zipState;
 %do i = 1 %to &N;
   %let filename = %sysfunc(compress(&&state&i));
   put '"c:\program files\winzip\wzzip"' "&filename" '.zip ' "&filename" '.csv';
 %end;
%mend zipState;

options nodate nonumber;
title;
data _null_;
 file 'c:\projects\wiasrd\zipstate.bat' print;
 %zipState;
run;
```

Here we use the PUT statement to produce all the command lines used by WinZip. One important thing is that the default title "The SAS System" and date could appear as the first line in the batch file. It will cause trouble when you execute the batch file because "The SAS System" is not a command at all. We use options to turn it off.

The batch file looks like:

```
"c:\program files\winzip\wzzip" Alabama.zip Alabama.csv
"c:\program files\winzip\wzzip" Alaska.zip Alaska.csv
"c:\program files\winzip\wzzip" Arizona.zip Arizona.csv
"c:\program files\winzip\wzzip" Arkansas.zip Arkansas.csv
. . .
```

STEP 5

To execute the batch file, you can locate it through Windows file explorer and then double-click it. The alternative way is to use X command in SAS.

```
options noxwait xsync;
x 'cd c:\projects\wiasrd';
x 'zipstate.bat';
```

If X command is used, you must be very careful with above statements. You may wonder why not use only one X statement like x 'c:\projects\wiasrd\zipstate.bat' instead of going to the directory and then executing the batch file. The reason is that with the one statement approach, it will go to the default SAS system user work directory such as C:\Documents and Settings\myaccount and then execute the batch file. It fails since it could not find the files to be zipped because they are in the different folder - C:\projects\wiasrd. With the two X statements, it guarantees the batch file and the files to be zipped under the same directory.

CONCLUSION

Processing the raw input data is the starting point for all later work steps. This paper provides a case example on how to use PUT _INFILE_ statement to write to external files. Also, it shows how to efficiently use third-party product in SAS. The principle is good for subsetting the geographical input data.

REFERENCES

SAS Institute Inc. (1999), *SAS Online Doc, Version 8*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author would like to thank Mr. John Baj for reviewing this paper and providing good comments.

CONTACT INFORMATION

Your comments and questions are always welcomed. Contact the author at:

Ben Xu
Center for Governmental Studies
Northern Illinois University
148 N 3rd St
De Kalb, IL 60115
Work Phone: (815)753-4790
Email: dxu@niu.edu