

Paper PO-127

Using SAS® Macros to Split a Data Set into Several Output Files

Imelda C. Go, South Carolina Department of Education, Columbia, SC

ABSTRACT

Generating output files from various subsets of a master data set is a common task. For example, a statewide master data set has records grouped by district and the task requires splitting the master data into an output file per district. In this example, PROC FREQ is used to generate a data set with all the unique district values in the master data set. A macro loop is used to process the data per district. For each loop iteration, macro parameters are assigned values based on the data per district. These values, which vary per district, were used to give unique names to each output file. Another macro discussed enables the programmer to assign the number of observations in a data set to a macro variable. This macro variable is useful for determining how many iterations will be executed in the macro loop mentioned above, which eliminates hardcoding the number of iterations in the macro loop. The form of the output files may vary depending on the programmer's needs and is specified within the macro loop.

The first example goes through a very basic form of the technique with only the simplest macro programming concepts: global macro variable and macros with parameters. The second example illustrates the use of a macro loop and CALL SYMPUT function.

Both examples will use the following sample master data, which contains data for districts A, B, and C. The variables in the data set are district ID, district name, school ID, and school name. The primary goal for both examples is to divide the master data file by district.

Obs	districtid	district	schoolid	school
1	0801	A	001	W
2	0801	A	102	X
3	2001	B	202	Y
4	2001	B	202	Y
5	3001	C	303	Z
6	3001	C	303	Z
7	3001	C	303	Z

The following line is for defining a global macro variable that contains an output directory's path that will be used in the macro defined below.

```
%let outputdir=S:\Alib\Assess2\Data\test;
```

EXAMPLE 1

The following macro named writefile has three macro parameters for the district ID (districtid), district name (district), and number of records (numrecords) per district. The macro starts with a DATA _NULL_ step that isolates the records with a particular district ID value in a WHERE statement. The records from that district are then written into a file. The location of the file is specified by the outputdir global macro variable. The file name contains the district id, district name, and the number of records in the file. In addition to dividing the data into a file per district, the macro also to writes PROC FREQ output into an RTF file by district. The RTF file is defined by the ODS RTF statement and given a name that contains the district ID and district name. The PROC FREQ output shows the number of records by school per district.

```

%macro writefile (districtid,district,numrecords);

    data _null_;
        set master (where=(districtid="%&districtid"));
        file "&outputdir\subset &districtid &district &numrecords records.txt" lrecl=592;
        put districtid 1-4    district $9.    schoolid 13-15 school $17;
    run;

    ods rtf file="&outputdir\sas output &districtid &district..rtf";

    proc freq data=master (where=(districtid="%&districtid"));
        tables districtid*district*schoolid*school/list missing;
        title 'Number of records in each school';
    run;

    ods rtf close;
%mend writefile;

```

This example uses PROC FREQ to generate a data set with all known districts in the master data set. The value of fields in the data file will be assigned to macro parameters required by macro writefile. The macro processing will be driven by the data in the master data set, which eliminates manual hardcoding of the macro calls.

```

proc freq data=master;
    tables districtid*district/list missing out=districtinfo;
    title 'Districts in the master file';
run;

```

Districts in the master file					
The FREQ Procedure					
districtid	district	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0801	A	2	28.57	2	28.57
2001	B	2	28.57	4	57.14
3001	C	3	42.86	7	100.00

```
proc print data=districtinfo;
```

Districts in the master file				
Obs	districtid	district	COUNT	PERCENT
1	0801	A	2	28.5714
2	2001	B	2	28.5714
3	3001	C	3	42.8571

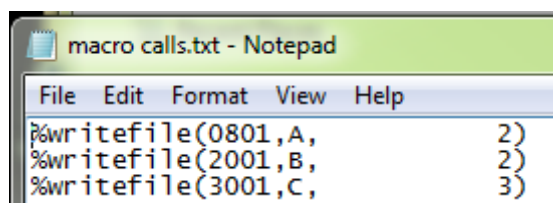
The data from the districtinfo output data set from PROC FREQ is used to create a text file with the macro calls with parameters per district. The following DATA _NULL_ step creates a text file in the output directory.

```

data _null_;
    file "&outputdir\macro calls.txt";
    set districtinfo;
    macrocall=%writefile('||districtid||','||trim(%str(district))||','||count||');
    put macrocall;
run;

```

The text file contains the following.



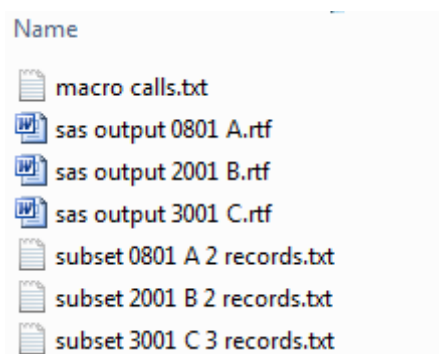
The file is then associated with a filename called maccall.

```
filename maccall "&outputdir\macro calls.txt";
```

```
%include maccall;  
run;
```

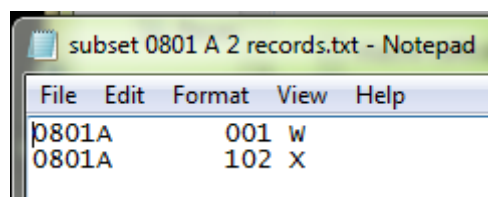
The %INCLUDE statement brings the statements in the maccall.txt file into the program for execution.

After program execution, the following files were written into the output directory:



The macro calls.txt file contains the %writefile macro calls used to produce the files per district.

The TXT files show the data files by district. Here is what the subset 0801 A 2 records.txt file contains:



Each RTF files shows the number of records per school within a district via the PROC FREQ output. Here is what the sas output 0801 A.rtf file contains:

Number of records in each school

The FREQ Procedure

districtid	district	schoolid	school	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0801	A	001	W	1	50.00	1	50.00
0801	A	102	X	1	50.00	2	100.00

EXAMPLE 2

The following statements from Example 1 are replaced by other statements to achieve the same results.

```
data _null_;
    *length macrocall $100.;
    file "&outputdir\macro calls.txt";
    set districtinfo;
    macrocall=%writefile('||districtid||','||trim(%str(district))||','||count||');
    put macrocall;
run;

filename maccall "&outputdir\macro calls.txt";

%include maccall;
run;
```

The new code consists of:

```
%macro obsnvars(ds);
    %global dset nob;
    %let dset=&ds;

    /* Open data set passed as the macro parameter */
    %let dsid = %sysfunc(open(&dset));

    /* If the data set exists, then grab the number of observations */
    /* then close the data set */
    %if &dsid %then
        %do;
            %let nob = %sysfunc(attrn(&dsid,nob));
        %end;
    /* Otherwise, write a message that the data set could not be opened */
    %else %put open for data set &dset failed - %sysfunc(sysmsg());
%mend obsnvars;

%obsnvars(districtinfo)

%macro writedata;
    %do index=1 %to %EVAL(&nob);

        data _null_;
            set districtinfo;
            if _n_=&index;
            call symput ('districtid',districtid);
            call symput ('district',district);
            call symput ('numrecords',count);
            %writefile (districtid,district,numrecords);

        %end;
    %mend writedata;

%writedata;
```

The %obsnvars macro uses a data set as its ds macro parameter. The macro will produce a global macro variable that contains the number of observations in the ds macro parameter. The %obsnvars(districtinfo) macro call assigns the number of variables in data set districtinfo to global macro variable nob.

The %writedata macro loops as many times as there are number of observations in districtinfo. Each loop is defined by the INDEX macro variable, which goes from 1 to the number of districts. The data for each district is accessed by the use of the subsetting IF statement (if _n_=&index;). The %writedata macro executes the %writefile macro per district with macro parameter values assigned by the CALL SYMPUT function.

The %obsnvars macro is described online at www.support.sas.com (KNOWLEDGE BASE / SAMPLES & SAS NOTES):

Sample 24671: Dynamically determine the number of observations and variables in a SAS data set

Details Full Code Output About Rate It

```
/* Create a test data set */

data test;
  input a b c $ d $;
datalines;
1 2 A B
3 4 C D
;

%macro obsnvars(ds);
  %global dset nvars nob;
  %let dset=&ds;

  /* Open data set passed as the macro parameter */
  %let dsid = %sysfunc(open(&dset));

  /* If the data set exists, then grab the number of observations and variables */
  /* then close the data set */
  %if &dsid %then
    %do;
      %let nob = %sysfunc(attrn(&dsid,nob));
      %let nvars=%sysfunc(attrn(&dsid,nvars));
      %let rc = %sysfunc(close(&dsid));
    %end;

  /* Otherwise, write a message that the data set could not be opened */
  %else %put open for data set &dset failed - %sysfunc(sysmsg());
%mend obsnvars;

/* Execute the macro -- pass TEST as the data set parameter */

%obsnvars(test)

%put &dset has &nvars variable(s) and &nob observation(s).;
```

REFERENCES

SAS Institute Inc. 2009. *Base SAS® 9.2 Procedures Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2010. *SAS® 9.2 Language Reference: Concepts, Second Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2011. *SAS® 9.2 Language Reference: Dictionary, Fourth Edition*. Cary, NC: SAS Institute Inc.

Sample 24671: Dynamically determine the number of observations and variables in a SAS data set. Retrieved July 17, 2015, from the SAS Support Web site: <http://support.sas.com/kb/24/671.html>

TRADEMARK NOTICE

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.