Macro Makes PROC MEANS Flexible
Kimberly D. Yarbrough, Bureau of the Census, Washington, DC

ABSTRACT

Have you ever had to aggregate a large data set (> 2 million records) by several variables? Did you have to vary the levels of aggregation based on a variables’ value in the source data set? Have you tried PROC MEANS/SUMMARY with a CLASS statement and realized that you had more than 100 levels of aggregates and had to figure out the _type_ values you actually need in your output data set? Or run out of memory performing this type of tabulation? This macro code allows you to predefine levels using %LET statements, output only the levels that you need and is flexible enough to output to more than 1 data set.

INTRODUCTION

Summarizing data for the 1997 Economic Census Core Business Statistics Publications provided an opportunity to expand on the PROC MEANS procedure. The requirements were to summarize a 5.5 million record data set by 9 categorical variables, creating summaries for four analysis variables. Your first inclination is to use PROC MEANS with a CLASS statement. However, the requirements are a bit more complicated. Our user requirements are to summarize 2 to all 4 of the analysis variables, by 1 to all 9 of the categorical variables and output the results to different data sets. The trick is to set up summary categories, based on values within the raw data set (See Figure 1), and write macro code to provide the flexibility.

Figure 1: Raw Data

<table>
<thead>
<tr>
<th>TRADE</th>
<th>SIC2</th>
<th>SIC3</th>
<th>SIC4</th>
<th>SIC6</th>
<th>ST</th>
<th>TX_TO</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>201</td>
<td>2011</td>
<td>201100</td>
<td>06</td>
<td>T</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>201</td>
<td>2012</td>
<td>201200</td>
<td>06</td>
<td>T</td>
<td>9000</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>302</td>
<td>3022</td>
<td>302211</td>
<td>05</td>
<td>N</td>
<td>5000</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>801</td>
<td>8011</td>
<td>801100</td>
<td>10</td>
<td>N</td>
<td>5000</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>802</td>
<td>8022</td>
<td>802211</td>
<td>43</td>
<td>N</td>
<td>3000</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>803</td>
<td>8033</td>
<td>803355</td>
<td>43</td>
<td>T</td>
<td>4000</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>803</td>
<td>8033</td>
<td>803355</td>
<td>43</td>
<td>T</td>
<td>5000</td>
</tr>
</tbody>
</table>

STEP 1: ASSIGNING SUMMARY LEVEL MACRO VARIABLES

First create macro variables for each of the predetermined set of categorical variables and assign values based on the trade’s summary requirements. In this example, there are 4 categories of tabs. Each summary requirement have all levels of SIC, where SIC6 is the most detailed summary and SIC2 the highest level of summary.

%macro disclose(dvar,trade,tradeval,area);
  data test1;
  set est.est97
  where &trade=&tradeval;
  if &trade='8' then do;
    /* Category 1: Trade/State/Taxable Indicator/SIC */
    %let st5=trade st tx_to;  
    %let st4=trade st tx_to sic2;  
    %let st3=trade st tx_to sic4;  
    %let st2=trade st tx_to sic4;  
    %let st1=trade st tx_to sic6;
  end;
  else if &trade in ('1','2','3','4','6','7') then do;
    /* Category 2: Trade/Taxable Indicator/SIC */
    %let t5=trade tx_to;  
    %let t4=trade tx_to sic2;  
    %let t3=trade tx_to sic4;  
    %let t2=trade tx_to sic6;  
    end;
  else if &trade in ('8') then do;
    /* Category 3: Trade/State/SIC */
    %let su5=trade st;
    %let su4=trade st sic2;
    %let su3=trade st sic3;
    %let su2=trade st sic4;
    %let su1=trade st sic6;
  end;
  else if &trade in ('8') then do;
    /* Category 4: Trade/SIC */
    %let u5=trade;
    %let u4=trade sic2;
    %let u3=trade sic3;
    %let u2=trade sic4;
    %let u1=trade sic6;
  end;
run;

Within each category the macro variables are numbered from one to five. The number of the macro variables is important in determining the order that the summaries are performed in the DO loop in Step 3. The number 1 represents the most detailed tab level and 5 represents the highest. This ensures in the DO loop that the most detailed data is tabbed first and then collapsed to a higher level in the next iteration.

STEP 2: INITIALIZING BASE DATA SETS

Two empty data sets are created to serve as the base tables in the PROC APPEND in Step 3. Our requirement was to create 2 data sets; one with U.S. level summaries and one with state level summaries.

data US&area;
  length &dvar 8. trade $1 tx_to sic2 $2 sic3 $3 sic4 $4 sic6 $6;
  set _null_; 
run;

data state&area;
  length &dvar 8. trade $1 tx_to st sic2 $2 st sic3 $3 st sic4 $4 sic6 $6;
  set _null_; 
run;

STEP 3: CREATING THE SUMMARIES

The TABS macro performs a PROC SORT, PROC MEANS and a PROC APPEND for each iteration of the DO loop. The parameter passed to the TABS macro determines which base...
data set the summary data is appended to. The data passed through the first iteration of the DO loop is the source data and is summarized at the lowest level of details. The intermediate summary is then appended to the base table and is passed back through the procedures and is aggregated to the next higher level. This continues until the highest level of summaries are performed and appended to the base table. Notice the use of the triple ampersands in all 3 procedures. The first scan of the macro reference resolves &d &i to &t1. The macro processor then rescans the text and resolves the reference to TRADE TX_TO SIC6 (Note: &t1 is assigned in Step 1).

```
%macro tabs(d);
data test&d.1;
  set test1;
  run;
  %do i=1 %to 5;
    %let j=%eval(&i+1);
    %if &d=t or &d=u  %then %do;
      proc sort data=test&d&i;
        by &&&d&i;
        run;
      proc means data=test&d&i noprint;
        var &dvar;
        by &d &d&i;
        id tx_to sic2 sic3 sic4 ;
        output out=test&d&j sum=;
      run;
    %end;
    %if &d=su or &d=st  %then %do;
      proc sort data=test&d&i;
        by &&&d&i;
        run;
      proc means data=test&d&i noprint;
        var &dvar;
        by &d &d&i;
        id tx_to sic2 sic3 sic4 ;
        output out=test&d&j sum=;
      run;
    %end;
  %end;
run;
%mend tabs;
```

The BY statement is used in the PROC MEANS rather than the CLASS statement because of insufficient memory errors. Several of the most detailed summary data sets are larger than 1 million records and were too large to process by 9 categorical variables.

**STEP 4: CALLING THE TABS AND DISCLOSE MACRO**

In this example there are 4 possible values to pass to the TABS macro; st (Category 1), t (Category 2), su (Category 3), and u (Category 4). The TABS macro is nested within the DISCLOSE macro and its parameters determine what categorical variables are used in the procedures. The

```
%if &tradeval='8' %then %do;
  %tabs(t);
  %tabs(st);
%end;
%if &tradeval='1' or &tradeval='2' or &tradeval='3' or &tradeval='4' or &tradeval='6' or &tradeval='7' %then %do;
  %tabs(u);
  %tabs(su);
%end;
%mend disclose;
%disclose(sr,trade,(8),8);
```

**CONCLUSION**

Macro language provides a great deal of efficiencies and flexibility in base SAS code. My example illustrates how macro language can be used in procedures to control the analysis variables, categorical variables, levels summarized and destination of a large number of summaries with very little code.

**CONTACT INFORMATION**

Kimberly D. Yarbrough
Senior Project Leader
Economic Planning and Coordination Division
Washington, DC 20233
kimberly.dee.yarbrough@ccmail.census.gov

SAS is a registered trademark of SAS Institute Inc, in the USA and other countries. ® indicates USA registration.