Paper RIV06

Instant Disaggregation: Using the macro language to provide reports with parallel structure across different subsets of the data set

Daniel Ralyea, Rock Hill Schools

ABSTRACT

SAS Macro Language automates repetitive processes. This allows standardization of reports across multiple locations and sub-groups. Two different techniques can be used to make parameter changes that distinguish the different reports. One technique is to read observations from a control table one at a time and use the parameters recorded in the control table this is similar to the process used by SAS Data Integration Studio ® for looping transformations. The second technique is to load the parameters in to an array style macro variable and call the parameters from a global macro variable table. This eliminates the need for repetitive opening of the control table. The use of these parameters with the dynamic variables in PROC SGRENDER allows for rapid coding of a wide variety of different graphs.

INTRODUCTION

Effective disaggregation of data is a primary tool in root cause analysis. SAS provides methods for rotating through a variety of filters that produce reports with the same structure applied to different subsets of the data. The parallel structure of the reports allows report consumers to quickly contrast the different subsets. When the macro language is combined with by group processing you can create easily navigated PDFs that allow report consumers to have reports customized by location and disaggregated by subgroups.

Overview: The flat file returned from a testing company has student scores that need to be presented as a whole group and disaggregated by location, grade level, ethnicity, socio-economic status (SES), English as a second language (ESL), and students with disabilities status (SWD). The expectation is that these reports will have the same basic structure to allow non-technical consumers to make observations contrasting the different subgroups.

The procedure to disaggregate the files requires falls into three steps. Preparing the data file, making sure that all the values for the subgroups to be reported on are populated. Preparing control tables, creating tables that identify the columns to be disaggregated. The use of control tables also prepares the program for use as a stored process utilizing prompts. Finally, creating the macro that produces the reports, using the macro variables to populate the dynamic variables in the ODS template.

PREPARING THE DATA FILES

The test file commonly is returned as a CSV file. SAS PROC IMPORT provides a strong utility for importing the data but the nature of the data may produce large numbers of observations with blank values (for instance a field indicating a student is homeless). PROC IMPORT will truncate these values unless the guessing rows parameter is set to a value large enough to read a row that contains a non-missing value.

```
PROC IMPORT OUT= work.testfile
    DATAFILE= "path\testfile.txt"
    DBMS=csv replace;
GETNAMES=yes;
DATAROW=2;
GUESSINGROWS=10000;
RUN;
```

Testing companies commonly publish a test file layout document. Abbreviations and codes are frequently used in the data files. The test layout will help to create formats to make the codes used in the CSV file descriptive for use in the report. The formatted values will be displayed in the graphs. The school codes used in the test file are usually tied to the school BEDS codes and not familiar to report consumers.

	LocationCode	StudentName	Gender	Ethnicity	SES	GTStatus	SPED	Grade_Level	Test_Raw	Test_Level
1	1	Student 1	M	W	P	S	9	5	635	HI
2	1	Student 2	F	B	N	N	3	4	595	LO
3	2	Student 3	M	A	P	N	9	4	610	AV

Figure 1. TestFile after Importing

```
PROC FORMAT;
VALUE $SchlID 1="School 1" 2="School 2";
VALUE $Gender "M" = "Male" "F"="Female";
VALUE $Ethnicity "W" = "Caucasian" "B" = "African American" "A" = "Asian";
VALUE $SES "P" = "Below Poverty Level" "N"="At or Above Poverty Level";
VALUE $GT "S"="State Identified" "L"="Locally Identified" "N"="Not Identified";
VALUE $SPED 3="Student with Disabilities" 9="Not Identified";
RUN;


PROC DATASETS LIBRARY=work;
MODIFY testfile;
FORMAT LocationCode $SchlId. Gender $Gender. Ethnicity $Ethnicity. SES $SES. GT $GT.
       SPED $SPED.;
QUIT;

PROC SORT DATA=work.testfile;
BY Grade_level;
RUN;
/* sort the dataset for by group processing with PROC SGRENDER */
```

PREPARING THE CONTROL TABLES

Because the data is being disaggregated by locations and subgroups two control tables will be generated. The first control table will contain the locations or schools. The second table will be a listing of the variables for disaggregation (using the variable names in the data file).

```
DATA locations;
INPUT locationcode   locationName $;
DATALINES;
1 School1
2 School2
;
RUN;

DATA groups;
INPUT subgroup $;
DATALINES;
Ethnicity
Gender
SES
SPED
ESOL
Grade_Level
;
RUN;
```

CREATING THE MACRO

Using macros allows control of system resources and program control commands while retaining the automation built-in to the DATA and PROC features of SAS. The basic flow of the program will be to read attributes of the control table then cycle through the values from the control tables, store the values as macro variables in memory then recall those values as text inside of a nested loop. The number of observations from each control table will provide the stop values in the nested DO loops. The values in the observations in the control tables will be passed as text into the code for the DATA Steps and ODS Graphic Design Layout. The outer loop will cycle through each location in the Locations table. Each observation in the Location table will be used once to subset the data for that set of graphs.

The inner loop will cycle through the values in the groups table so the observations in the Groups table will be used once for each observation in the Locations table.

```
%MACRO LocationData(); /*Start Macro Definition*/

%GLOBAL nobslocation;
/* %GLOBAL tells SAS to store the variables in memory */
/* that will remain available for later use otherwise the */
/* memory is cleared when the macro is finished running */

%LET SetId=%SYSFUNC(OPEN(Locations));

/* %LET is used to assign macro variables */
/* %SYSFUNC references system functions */
/* SetId is a number being associated with the */
/* dataset Locations further commands will use this */
/* number to reference the dataset. The OPEN command loads the */
/* Locations table into memory */

%LET nobslocation = %SYSFUNC(ATTRN(&SetId.,NOBS));

/* ATTRN returns a numeric attribute from the metadata of the dataset */
/* NOBS number of observations in the dataset */

%LET rc=%SYSFUNC(CLOSE(&Setid.));
/* The CLOSE command releases the dataset from memory */
%MEND;
%LocationData();
```

The same basic steps will return the number of observations in the group table and load the subgroup values in memory. Storing the subgroups in the global resource table uses more memory but reduces processing time when compared to reloading the table for the iterations of the inner loop.

```
%MACRO SubgroupData();

/*Start Macro Definition*/

%GLOBAL nobsgroups SubGroupNameCol;
%LET SetId=%SYSFUNC(OPEN(Groups));
%LET nobsgroups = %SYSFUNC(ATTRN(&SetId.,NOBS));

/* the macro variable nobsgroups stores the number of observations */
/* in the groups table*/

%DO i=1 %TO &nobsgroups;

/* &nobsgroups is a text string that contains the digits stored */
/* by the %let statement */

%GLOBAL subgroup&i. ;

/* This adds a variable to the global resource table called */
/* subgroup1 on the first pass */
/* subgroup2 will be created on the second pass . . . until */
/* there is a macro variable created */
/* to store each subgroup name */

%LET rc =%SYSFUNC(FETCHOBS(&SetId.,&i.));

/* loads observation number i into PDV (program data vector) */

%LET subgroup&i. = %SYSFUNC(GETVARC(&SetId.,%SYSFUNC(VARNUM(&SetId.,Subgroup))));

/* %SYSFUNC(VARNUM(&SetId.,Subgroup))returns the position of the */
/* variable Subgroup in the PDV */
/* GETVARC returns the character value stored in the dataset */
/* numbered SetId in the position returned by the VARNUM function. */
/* The character value is stored in the global macro variable */
/* named subgroup&i. when referencing the macro variable subgroup&i */
/* two &s must be placed in front to tell the compiler to */
/* take multiple passes (2) to resolve this variable.*/

%PUT &&subgroup&i.;

/* Prints the values stored in the macro variable subgroup&i  to the log*/

%END;
%LET rc=%SYSFUNC(CLOSE(&Setid.));

/* releases the dataset from memory*/

%MEND;

%SubgroupData();
```

These two macros have stored several variables in memory.

Nobslocation- the number of observations in the location table

Nobsgroups- the number of observations in the groups table

Subgroup1-Subgroup(n)- the subgroup names by which the data is to be disaggregated

The style of the graphs being produced is determined using PROC TEMPLATE. The PROC TEMPLATE portion of the macro was generated using ODS Graphic Designer. Taking a simple bar graph prepared using sample data and replacing the variables from the sample data with dummy variables that will be assigned from the macro

variables generated earlier. The difference between the dynamic variables used in the template procedure and the macro variables is when the variables are resolved. Macro variables are resolved when the specific iteration is compiled and dynamic variables are resolved at execution. PROC TEMPLATE will only need to be compiled and run once defining the overall behavior of the graph. The macro variables will be resolved during each iteration of the loop and used in the PROC SGRENDER to indicate the variables in the data set to be graphed.

```
PROC TEMPLATE;
DEFINE statgraph GraphA;

/* defines the type of template being produced as a statistical*/
/* graph named GraphA */

DYNAMIC _var1 _var2;
/* names two variables as dynamic variables that will be resolved*/
/* when the template is called */

BEGINGRAPH;
LAYOUT lattice / ROWDATARANGE=data COLUMNDATARANGE=data ROWGUTTER=10
COLUMNGUTTER=10;

/* Organizes the page for graph placement */

LAYOUT overlay;
BARCHART X=_var1 Y=_var2 / NAME='bar' STAT=mean GROUPDISPLAY=Cluster
CLUSTERWIDTH=.5;

/* Defines graph overlay */

ENDLAYOUT;
ENDLAYOUT;
ENDGRAPH;
END;
RUN;
```

The final step is to use the information gathered in the earlier macros with the template that has been defined to produce the graph sets. These graphs will be produced by ODS as PDFs.

```
ODS HTML CLOSE;

/* Close the HTML destination */

GOPTIONS RESET=all;

/* Reset the graphics options */
OPTIONS
PAPERSIZE=A4
LEFTMARGIN=1cm
RIGHTMARGIN=1cm
BOTTOMMARGIN=1cm
TOPMARGIN=2cm;
ODS GRAPHICS ON;
/* Modify the PDF page properties. These options must */
/* be set before the ODS PDF statement is executed. */
```

Since the goal is to produce sets of graphs by location the ODS invocation will be placed in the macro. The outer loop of the macro will run through the following steps. The LocationCode and LocationName will be retrieved from the Locations dataset and be held in memory as macro variables. The LocationCode will be used to subset the test dataset. The LocationName will name the file and title the graphs. The inner loop will cycle through the subgroups, pass the subgroup variables through to and execute the PROC SGRENDER.

```
%MACRO PrintGraph();

/*Start Macro Definition*/

%LET SetId=%SYSFUNC(OPEN(Locations));

/* Open Locations dataset */

%DO j=1 %TO &nobslocation;

/* Use the number of locations that was stored in the global variables to */
/* set the loop boundary */

%LET rc =%SYSFUNC(FETCHOBS(&SetId.,&j.));

/* Load in the jth observation of the Locations dataset into the data vector */

%LET LocName = %SYSFUNC(GETVARC(&SetId.,%SYSFUNC(VARNUM(&SetId.,locationname))));
%LET LocID = %SYSFUNC(GETVARC(&SetId.,%SYSFUNC(VARNUM(&SetId.,locationcode))));

/* Store the Location Name and Location Code in a local macro variable */


ODS PDF STYLE=seaside
FILE="C:\Documents\Reports\&LocName..pdf";

/* Open the PDF destination and set the output file name */
/* based on the Location Name */

%DO i=1 %TO &nobsgroups.;

/* Loop through the subgroups stored in the global macro variable table */

TITLE1 “Test Score Means for &LocName. grouped by="&&subgroup&i.";

PROC SGRENDER DATA=work.pass TEMPLATE=Graph; WHERE LocationCode eq &LocID.;
DYNAMIC _var1="Test_Raw" _var2="&&subgroup&i.";
RUN;

TITLE1 “Test Score Means for &LocName. paged by Grade_Level grouped
by="&&subgroup&i.";

PROC SGRENDER DATA=work.pass TEMPLATE=Graph; WHERE LocationCode eq &LocID.; BY
Grade_Level;
DYNAMIC _var1="Test_Raw" _var2="&&subgroup&i.";
RUN;

TITLE1 ;

/* Produce the graphs passing the Location code to subset the testing table and */
/* finding and graphing the mean by the subgroup stored in &&subgroup&i. */
/* The second PROC SGRENDER uses by group processing to further disaggregate */
/* each subgroup by grade_level */
%END;
ODS PDF CLOSE;
%END;
%LET rc=%SYSFUNC(CLOSE(&Setid.));
%MEND;
%PrintGraph();
```

CONCLUSION

The techniques reviewed can be extended by building different parameters in the control tables beyond the listing of variables or school names. For example including distinct paths for each school can distribute the graphs through the network or using email addresses to notify consumers that the reports are complete. (The reports themselves are not emailed because of privacy/security concerns with email) The table used for this paper was simplified for clarity. The original test files contain several scores spanning multiple subjects. Each PROC SGRENDER produces a node in the PDF that allows report consumers to navigate through the sets of graphs.

REFERENCES

Carpenter, Art. 2004. Carpenter's Complete Guide to the SAS® Macro Language, Second Edition. Carey, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Daniel Ralyea
Rock Hill Schools
660 N. Anderson Rd
Rock Hill, SC 29730
(803) 981-1050
DRALYEA@RHMAIL.ORG