UISUG SAS Code Swap

April 7, 2010-04-07

```
/************************************************************/
/*  MACRO TO CREATE MULTIPLE SAS DATASETS FROM ONE SAS        */
/*  DATASET BASED UPON THE VALUE OF THE BY-GROUP VARIABLE     */
/*  This program code creates a SAS macro that can be used to generate  */
/*  a new dataset for each BY-group in an existing data set.  A detailed  */
/*  discussion of this code can be found in SAS Institute Sample 26140.  */
/************************************************************/

/*  Create sample data                                        */

data test;
 input color $ num;
datalines;
blue 1
blue 2
blue 3
green 4
green 5
red 6
red 7
red 8
;

/*  Create a new macro variable, VARn, for each BY-Group and a counter  */
/*  of the number of new macro variables created.             */

data _null_;
  set test end=eof;
   by color;

/*  On the first member of the BY-Group, create a new macro variable  */
/*  VARn and increment the counter FLAG.                      */

if first.color then do;
  flag+1;
  call symput('var'||put(flag,8. -L),color);
end;

/*  On the last observation of the data set, create a macro variable to  */
/*  contain the final value of FLAG.                          */

if eof then call symput('tot',put(flag,8. -L));
run;
```

```
/*  Create a macro to generate the new data sets. Dynamically produce    */
/*  data set names on the DATA statement, using subsetting criteria to   */
/*  create the new data sets based upon the value of the BY variable.    */

%macro groups(dsn,byvar);
   data %do I = 1 %to &tot;
     &&var&i
     %end;;
   set &dsn;
     %do I = 1 %to &tot;
     if &byvar = "&&var&i" then output &&var&i;
    %end;
run;
%mend groups;

/*  Call the macro GROUPS.  Specify the name of the data set to be split  */
/*  in the first macro parameter and the name of the BY variable in the   */
/*  second parameter.                                                     */

%groups(test,color)
proc print data = blue;
  title 'Blue Group';
run;

proc print data = green;
  title 'Green Group';
run;

proc print data = red;
  title 'Red Group';
run;
```

UISUG SAS Code Swap

April 7, 2010-04-07

```
/**************************************************************/
/*  MEAN SUBSTITUTION FOR MISSING VALUES                      */
/*  This program code provides an easy way to replace missing values   */
/*  in a dataset with its mean value for multiple variables at the same */
/*  time.  It uses a little known SAS procedure called PROC STANDARD   */
/*  to standardize some or all of the variables and create a new SAS   */
/*  dataset that contains variable means and/or standard deviations.   */
/*  In addition, there is a REPLACE option that substitutes the variable */
/*  mean for each of the missing values.                       */
/**************************************************************/
/*  The following SAS code demonstrates the use of PROC STANDARD    */
/*  for mean substitution.                                     */

data raw;
  input v1-v10;
datalines;
1 1 1 1 1 . 1 1 1 1
2 2 2 . 2 . 2 2 2 2
3 3 3 3 3 3 . . 3 3
4 4 4 . . 4 4 4 4 4
5 5 5 5 5 5 5 5 . .
;
proc standard data = raw out = stnd replace print;
  var v1-v10;
run;


/*  This SAS code demonstrates another way of substituting mean values  */
/*  for missing values.                                        */

data raw;
  input v1-v10;
datalines;
1 1 1 1 1 . 1 1 1 1
2 2 2 . 2 . 2 2 2 2
3 3 3 3 3 3 . . 3 3
4 4 4 . . 4 4 4 4 4
5 5 5 5 5 5 5 5 . .
;

/*  Use PROC MEANS to produce a new dataset meandat which has      */
/*  variables m1 through m10 holding the means for the variables v1 */
/*  through v10. PROC PRINT is used to verify this.            */

proc means noprint;
  var v1-v10;
```

```
   output out = meandat (drop = _type_ _freq_) mean = m1-m10;
run;

proc print data = meandat;
run ;

/*  This DATA step performs the variable substitution, creating a final         */
/*  dataset called meansub.  It defines two arrays: old represents v1           */
/*   through v10 and means represents m1 through m10.  A DO loop                */
/*   moves through the array variables, checking each value of array old        */
/*   to see if it is missing.  If value is missing, then the value is set to the */
/*   corresponding value from the array means.                                  */

data meansub (drop = m1-m10 i);
   if _n_ = 1 then set meandat;
   set raw;
   array old(10) v1-v10;
   array means(10) m1-m10;
   do i = 1 to 10;
      if old(i) = .  then old(i) = means(i);
   end;
run;
```

This program will run code for each site listed in a dataset.

```
LIBNAME DATA 'your path';
%LET SITELIST=DATA.SITELIST;  dataset that lists the sites of interest (each row=one site)
                               variable RECNUM enumerates sites in SITELIST
%LET DSID=%SYSFUNC(OPEN(&SITELIST)); function OPEN returns an internal pointer to SITELIST
%LET NSITES=%SYSFUNC(ATTRN(&DSID,NLOBS)); determining how many sites in SITELIST
%LET DSID=%SYSFUNC(CLOSE(&DSID));

%MACRO SITE(ID,NAME); this macro contains code to run for all sites in SITELIST

 your code

%MEND SITE;

%MACRO SITEINFO; the macro takes site ids and names from SITELIST and calls macro SITE
%DO I=1 %TO &NSITES;
   PROC SQL NOPRINT;
     SELECT SITEID INTO :SITEID FROM &SITELIST WHERE RECNUM=&I;
     SELECT SITENAME INTO :SITENAME FROM &SITELIST WHERE RECNUM=&I;
   QUIT;
   %SITE(&SITEID,&SITENAME);
 %END;
 %MEND SITEINFO;

 %SITEINFO

 RUN;
```
Questions? E-mail elena-perkhounkova@uiowa.edu

This code helps save and organize programs, logs, and outputs during program run.

```
1 DM 'LOG;CLEAR;OUT;CLEAR;';    /*CLEARING LOG & OUTPUT WINDOWS */

2 %PUT &SYSDATE &SYSTIME;       /*PUT DATE & TIME ON LOG        */ compare to line 6

3 %LET CURTIME=%SYSFUNC(TIME(),TIMEAMPM11.0);

4 %LET CURDATE=%SYSFUNC(TODAY(),WORDDATE.);

5 FOOTNOTE "THIS OUTPUT WAS CREATED AT &CURTIME ON &CURDATE"; /*ADDS FOOTNOTE TO OUTPUT*/

6 %PUT THIS PROGRAM RAN AT &CURTIME ON &CURDATE; /*ADDS TIME AND DATE OF PROGRAM RUN TO LOG*/


7 FILENAME LOG     '/your path/project.log';

8 FILENAME OUT     '/your path/project.out';

9 FILENAME PGM     '/your path/project.sas';


10 DM 'PGM;RECALL;FILE PGM REP;'; /*SAVE PROGRAM AS RUN*/


your code


11 RUN;
12 *';*";*/; to close all unclosed quotation marks

13 DM 'LOG;FILE LOG REP;'; /*SAVE LOG AS RUN*/

14 DM 'OUT;FILE OUT REP;'; /*SAVE OUTPUT AS RUN*/
```

Questions? E-mail elena-perkhounkova@uiowa.edu

```
/* uisug_codeswap_mckirgan_20100407.sas      */
/*********************************************/
/* UISUG April 2010 Meeting -- SAS Code Swap */
/* Bill-McKirgan@uiowa.edu                   */
/*********************************************/

/* make a datestamp for use as file name suffix
   where the date is YYYYMMDD (no dashes, hyphens, just numbers)
   this is handy for naming routine report files so they are
   kept in a group by report name followed by year, month and day

   i started using this after racking up several years of report
   data files and struggling to find a reference I needed later
   with everything grouped by Report name, then month, then day and year.
*/

%let datestamp=%sysfunc(today(),yymmddN8.);

%put &datestamp.;









/* get rid of all labels in a dataset .... sometimes they just get in the way
   if i'm browsing data while writing code those labels can get in the way
*/

data zipcode_&datestamp. ; set sashelp.zipcode;

attrib _all_ label ='';

run;
```

Hash objects

For this particular example hash objects are not the most efficient method, but this is a simple example of how to use hash objects to perform a function similar to the vlookup formula in excel:

```
DATA WORK.TARGET;
   INPUT UNIQUEID sex $ age group $ test1;
   DATALINES;
 1  F  35 A 17
17  M  50 A 14
33  F  45 B  6
49  M  24 E 14
65  F  52 D  9
81  M  44 B 11
2   F  34 C 17
18  M  40 C 14
34  F  47 A  6
50  M  35 E 17
;
DATA WORK.SOURCE_DOC;
      INPUT CLASS $ SCORE ;
   DATALINES;
A   20
B   19
C   10
D   14
;  /*NOTICE I HAVE NOT LISTED A SCORE FOR CLASS 'E' SO THAT YOU CAN SEE WHAT HAPPENS WHEN THERE IS NOT A MATCH
(RC≠0)*/
RUN;
DATA WORK.ONE (DROP = score class);
      IF _N_=1 THEN DO;
            IF 0 THEN SET WORK.SOURCE_DOC;
            DECLARE HASH VLOOKUP    (DATASET: "WORK.SOURCE_DOC")  ;
                              /*WORK.SOURCE_DOC IS THE DATA SET WHERE THE VALUES WE WANT TO LOOKUP ARE KEPT, WE
                              WILL LOOK THEM UP FROM THIS TABLE USING A UNIQUE IDENTIFIER COMMON TO BOTH DATA
                              SETS.  'VLOOKUP' IS THE NAME OF THE HASH OBJECT.
                              YOU CAN CHANGE THIS NAME TO ANYTHING*/
                  VLOOKUP.DEFINEKEY  ("CLASS") ;      /*'CLASS' IS THE VARIABLE IN THE SOURCE DATA SETS TO BE
                                    USED TO LOOKUP THE VALUES WE ARE LOOKING FOR, IT CAN HAVE A DIFFERENT NAME IN
                                    THE DESTINATION DATA SET*/
                  VLOOKUP.DEFINEDATA  ("SCORE") ; /*'SCORE' IS THE VARIABLE NAME FOR THE VALUES WE ARE LOOKING UP
                                          BASED ON UNIQUEID*/
```

Hash objects

```
                VLOOKUP.DEFINEDONE ( ); /*YOU NEED THIS LINE, BUT I CAN'T REMEMBER WHY*/
        END;

        SET WORK.TARGET ; /*WORK.TARGET IS THE DATA SET WHERE WE NEED TO ADD THE 'SCORE' VALUES*/
        RC=   VLOOKUP.FIND(KEY:GROUP); /*ID IS THE VARIABLE NAME IN THE DATA SET WORK.TARGET THAT CONTAINS
                        THE UNIQUE IDENTIFIERS WHICH WILL MATCH WITH SOME OR ALL OF THE UNIQUE IDENTIFIERS
                        IN THE WORK.SOURCE_DOC DATA SET.*/
            IF RC=0 THEN DO;  /*RC=0 WHEN THE VALUE FOR THE VARIABLE 'CLASS' IN WORK.SOURCE_DOC IS
                                THE SAME AS THE VALUE FOR THE VARIABLE 'GROUP' IN THE WORK.TARGET DATA
                               SET*/
                TEST2 =SCORE; /*TEST2 IS THE VARIABLE NAME WE WILL ASSIGN TO THE SCORE VALUES THAT WE
                                  HAVE LOOKEDUP AND ADDED TO THE WORK.TARGET DATA SET AND SAVED TO THE
                               WORK.ONE DATA SET*/
            END; ELSE DO;  TEST2=".";    END;

RUN;
```

THE RESULTING DATA SET WORK.ONE:

The SAS System

UNIQUEID	sex	age	group	test1	RC	TEST2
1	F	35	A	17	0	20
17	M	50	A	14	0	20
33	F	45	B	6	0	19
49	M	24	E	14	160038	.
65	F	52	D	9	0	14
81	M	44	B	11	0	19
2	F	34	C	17	0	10
18	M	40	C	14	0	10
34	F	47	A	6	0	20
50	M	35	E	17	160038	.

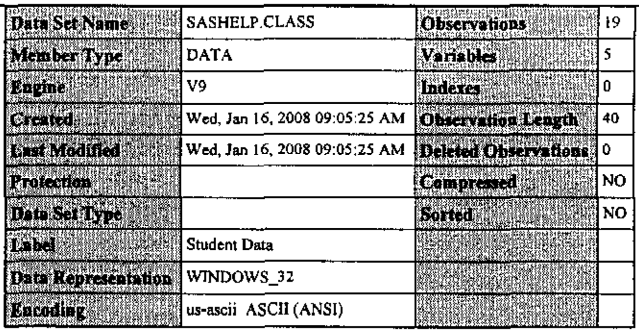
(NOTE: THE TABLE ABOVE WAS CREATED BY RIGHT-CLICKING ON THE DATA SET IN THE EXPLORER WINDOW OF SAS AND SELECTING 'VIEW IN EXCEL,' THEN I PASTED THE TABLE TO THIS DOCUMENT.)

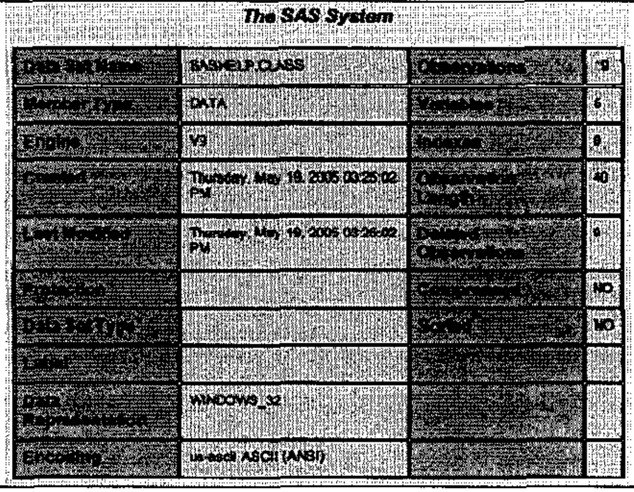
/*NOTE:YOU CAN LOOKUP VALUES FOR SEVERAL VARIABLES IN THIS DATA STEP BY ADDING THEM TO THE LIST IN THE DEFINEDATA COMMAND AND THEN CREATING SIMILAR IF-THEN-DO-ELSE LOOPS FOR EACH VARIABLE.*/

SAS output to microsoft word, html, or pdf files

Reference: http://support.sas.com/rnd/base/ods/scratch/ods-from-sc-paper.pdf
To easily print (somewhat) attractive SAS output in a htmL, pdf, or Microsoft Word document, you can use the following code. For example purposes I have included a proc contents procedures. Repace that line with any procedures you want to run. For all these examples you may want to include a filepath before the filename. (*e.g.* "c:\documents and settings\sas files\myreport.rtf")

OUTPUT DESTINATION (TYPE OF FILE)	Example code	**Output looks like**

Microsoft Word (RTF)	`ods rtf;`
	`proc contents data=sashelp.class;`
	`run;`
	`ods rtf close;`
	
	you will be given the option to open or save the file
	
	`OR:`
	
	`ods rtf file='myreport.rtf';`
	`proc contents data=sashelp.class;`
	`run;`
	`ods rtf close;`
		

HTML	`ods html;`
	`proc contents data=sashelp.class;`
	`run;`
	`ods html close;`
	
	`/*or to save the file: */`
	`ods html file="myreport.html";`
	`proc contents data=sashelp.class;`
	`run;`
	`ods html close;`
		

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Wed, Jan 16, 2008 09:05:25 AM	Observation Length	40
Last Modified	Wed, Jan 16, 2008 09:05:25 AM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

The SAS System

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Thursday, May 19, 2005 03:25:02 PM	Observation Length	40
Last Modified	Thursday, May 19, 2005 03:26:02 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		