SAS: Useful Tips

η ομάδα
August 14, 2013

SUMMARY

This is a collection of useful SAS ® code examples that are a result of practical experience and applied problem solving. Most of the programs involve either making the SAS *program* work in a more efficient way, or modifying SAS *code* to do what you want *with the data* with which you are working.

This document will detail each specific problem that has been encountered, the solution to the problem, and the SAS code to carry out the solution.

SAS CODE AND EXAMPLES

The code that is presented in this document can be typed (or copied and pasted) into the SAS system. However, the code is also available on the UISUG website (available here) for future reference.

1 Problem: Code and Data organization

Many times, a data analyst works on multiple projects at the same time. Also, this person may also be asked to assist with "quick turn-around" data analysis requests. Having the data organized by person or task is an easy way to keep track of what data is in the SAS system, and which results or output are related to which tasks.

1.1 Scenario

A person has asked you to analyze some data that is saved as a `.csv` file. The data is related to a cancer study, and the data is outside of the regular data sets that you normally use in your daily work.

You will have to import the data into the SAS system, and work with the data. What has to occur for this to happen is that the SAS program has to know where the data is located, and know how to reference the data for analysis. The way SAS does this is to read the data into a *library*. The library will be used to reference the **actual, physical** location of the data. Once SAS knows where the data is, and what it is called, SAS can work with it.

For the following example, suppose you want to organize the library by the person you are helping, and the data by the content. In this case, it is outcome data from a cancer study. Also, suppose the person's name is Thomas Anderson, and the file is named `results.csv`.

1.2 Code

```
libname Anderson '[location on computer]' ;
proc import file = '[location on computer]\results.csv' out=anderson.cancer;
getnames=yes; datarow=2;
run;
```

Another way to approach this scenario would be to use a little bit of macro variable assignment and only have to change one line of code for every time it is used. Like this:

```
%let folder_location = H: ;
libname Anderson = "&folder_location" ;
proc import file = "&folder_location.\results.csv" out=anderson.cancer;
getnames=yes; datarow=2;
run;
```

1.3 OUTCOME

Three things will happen when code like this is run:

1. The base file (the `.csv` file) will remain the same
2. A new SAS library will be created (in this case, named Anderson)
3. The new library contains the data as a SAS formatted file (so it can be analyzed)

Any analysis to be done on the data will have to reference the data using the SAS location, in this case, `anderson.cancer`.

2 Problem: Sharing Data for an Example

Often, you will want to share some data or demonstrate something in SAS, but it is sometimes difficult to send the *code* along with the *data* that will be used by the code. The good news is that for small examples, it is possible (and generally preferred) to include the data *within* the code.

2.1 Scenario

You are demonstrating a SAS procedure or programming issue, and you want to include a dataset that everyone can use without having to deal with importing the data from an outside format. This can be done on a small scale by including the data at the head of the program by using the `input` and `cards` options.

2.2 Code

The following code creates the dataset `measures` in the `work` folder.

```
data measures;
input f_name $10. sex $  age height   weight ;
cards;
Alfred      M      14     69.0      112.5
Alice       F      13     56.5       84.0
Barbara     F      13     65.3       98.0
Ronald      M      15     67.0      133.0
Thomas      M      11     57.5       85.0
William     M      15     66.5      112.0
run;
```

2.3 Outcome

When this code is run by SAS, a dataset is created in the work folder, and it can then be analyzed or manipulated by the SAS system.

3 PROBLEM: SORTING ITEMS BY LINGUISTIC RULES

It is possible that you may have data that needs to be sorted, but not using the ASCII encoding rules, but by using "language" rules instead. This sorting problem occurs mostly when numbers are included but are not the same length (e.g., putting the number 10 ahead of the number 7).

3.1 SCENARIO

You have a variable in a dataset that relates the order of a section of a paper to the number of that section. However, the numbers will not sort correctly. You will have to sort the variable `parts` by using the `linguistic` option in SAS.

3.2 CODE

The following code creates the dataset and demonstrates the incorrect and correct way to sort the data.

```
data bf_sort;
input parts $12.;
cards;
10.2.f.ii
10.2.a.iii
10.2.f.i
10.2
8.4.5.b.i
10.2.c.iii
10.2.d.i
10.2.d.ii
8.4.5.c.i
10.2.e.ii
10.2.e.i
7.5.2.b
7.5.2.c
8.4.5.e.i
10.2.b.iii
run;


*****  does NOT produce correct order ;

proc sort data=bf_sort out = bad_sort;
by parts;
run;



*****  does produce correct order ;

proc sort data=bf_sort sortseq = linguistic(numeric_collation=on) out = good_sort;
by parts;
run;

proc print data = bad_sort ;
run;

proc print data = good_sort ;
run;
```

3.3 Outcome

Below, you can see snippets of the incorrect sort and correct sort:

Bad Sort	Good Sort
10.2	7.5.2.b
10.2.a.i	7.5.2.c
10.2.a.ii	7.5.2.d
10.2.a.iii	7.5.3.a
10.2.b.i	8.4.5.a
10.2.b.ii	8.4.5.b.i
....	....
7.5.2.c	10.2.d.i
7.5.2.d	10.2.d.ii
7.5.3.a	10.2.d.iii
8.4.5.a	10.2.e.i
8.4.5.b.i	10.2.e.ii
8.4.5.c.i	10.2.e.iii
8.4.5.d.i	10.2.f.i
8.4.5.e	10.2.f.ii
8.4.5.e.i	10.2.f.iii

4 PROBLEM: RENAMING VARIABLES

SAS has some shortcuts in programming that are available to users when certain conditions are met. Often, one of those conditions is that variables need to have the same prefix, but can have a different suffix (e.g., `part01, part02` – `part38`, etc.). One of the shortcuts is to use the colon operator on a variable name. When the colon operator is used, it indicates that the SAS system should include all variables (or other things) that have the same prefix. For example, if we want to get the summary statistics for all the variables in a survey, we can name them all with the same prefix `qn` and use the colon operator to tell SAS to use all the variables with the prefix `qn` by using `qn:` .

4.1 SCENARIO

You have data that is comprised of several variables, and you would like them to all be named with the same prefix. This can be accomplished by using an `array` statement, and re-assigning each member of the array the new prefix .

The following code creates a datatable named `toons`, with four variables: `yakko`, `wakko`, `dot`, `buttons`, and `mindy` [1]. In order to use the array, the variables have to be adjacent and in the order that you want to number them. Suppose you want to rename the variables `yakko`, `dot`, and `mindy` to `toon1`, `toon2`, and `toon3`. First, you have to keep the variables in the order that you want. Second, you have to specify the new prefix in an `array` statement.

4.2 CODE

```
data toons;
input ID $4. yakko wakko dot buttons mindy;
cards;
0001 0 1 1 2 3
0002 2 2 1 2 0
0003 3 4 4 3 2
0004 1 0 0 2 1
0005 4 3 4 4 4
;
run;
```

[1] Quite possibly, the best example data ever.

```
*****  keep and order the variables as desired ;

data toons_reorder;
retain id yakko dot mindy;
set toons;
keep id yakko dot mindy ;
run;

*****  rename the variables using an array  ;

data toons_renamed (keep=ID toon1 -- toon3);
set toons_reorder;

array old{3} yakko -- mindy;
array toon{3} ;

do i=1 to 3;
toon[i] = old[i];
end;
run;
```

4.3 OUTCOME

The outcome of the statements is a dataset with four variables, `ID`, and three variables named `toon1, toon2,` and `toon3`. With the names having the same prefix, you can type your code a little more easily. For example, try:

```
data tmp ; set toons_renamed;
toon_sum = sum(of toon: ) ;
run;

proc means data = tmp;
var toon: ;
run;
```

5 PROBLEM: COUNTING VALUES THAT DO NOT EXIST

In SAS, it is sometimes difficult to produce frequencies of values that do not exist. For example, you may want to calculate demographic information variables like: `race` and `ethnicity`, however, if there are no occurrences of a variable, the frequencies would not be produced. This is an issue if the frequenies of values that have to be reported to satisfy some governing body reporting requirement (e.g., Federal agencies).

5.1 SCENARIO

You need to report the demographics of the participants in your study. There are no Hispanic participants so far, but you need to display a sex by ethnicity table and race by ethnicity table. SAS can do this in `PROC REPORT` by declaring the levels of the variables of interest, regardless of whether they exist in the dataset.

5.2 CODE

```
proc format;
      value race  1 = 'American Indian/Alaska Native'
                  2 = 'Asian'
                  3 = 'Native Hawaiian or Other Pacific Islander'
                  4 = 'Black or African American'
                  5 = 'White'
                  6 = 'More Than One Race'
                  7 = 'Unknown or Not Reported';

      value eth   1 = 'Hispanic or Latino'
                  2 = 'Not Hispanic or Latino'
                  3 = 'Unknown or Not Reported';

      value sex   1 = 'Male'
                  2 = 'Female'
                  3 = 'Unknown or Not Reported';
run;
```

```
data race;
input race  eth  sex   ;
cards;
1   2   1
1   2   1
1   2   2
2   2   1
2   2   2
2   2   1
3   2   2
3   2   2
3   2   1
4   2   2
4   2   1
4   2   2
run;

***** Does not work correctly ;

proc tabulate data = race;

class eth sex ;
format eth eth. sex sex.  ;
table (eth all = 'Ethnicity Category: Total of All Subjects'),
sex all = 'Total'

            / printmiss misstext='0';

      title 'No Format - Hispanic is Missing';
run;

***** Does work correctly ;

proc tabulate data = race;
format eth eth. sex sex.  ;
class eth sex / preloadfmt;

table (eth all = 'Ethnicity Category: Total of All Subjects'),
sex all = 'Total'

            / printmiss misstext='0';

      title 'Formatted - Hispanic is Present';
run;
```

5.3 OUTCOME

With the specification of `format` and `preloadfmt` in the second `PROC TABULATE` statement, the value of Hispanic is shown in the frequency table, even though there are no Hispanic participants in the study so far.

```
***** !!!!! table without preloadfmt  ;
------------------------------------------------------------
-                      -           sex           -           -
-                      ---------------------------           -
-                      -    Male    -   Female   -   Total   -
-                      ----------------------------------------
-                      -     N      -     N      -     N     -
------------------------------------------------------------
-eth                   -            -            -           -
-----------------------            -            -           -
-Not Hispanic or Latino-       6.00 -       6.00 -      12.00 -
------------------------------------------------------------
-Ethnicity Category:   -            -            -           -
-Total of All Subjects -       6.00 -       6.00 -      12.00 -
------------------------------------------------------------
```

```
***** !!!!! table with preloadfmt ;
------------------------------------------------------------------------
-                      -                sex                  -           -
-                      ---------------------------------------           -
-                      -           -           - Unknown or -           -
-                      -    Male   -   Female  -Not Reported-   Total   -
-                      ---------------------------------------------------
-                      -     N     -     N     -     N     -     N     -
------------------------------------------------------------------------
-eth                   -           -           -           -           -
-----------------------           -           -           -           -
-Hispanic or Latino    -         0 -         0 -         0 -         0 -
------------------------------------------------------------------------
-Not Hispanic or Latino-      6.00 -      6.00 -         0 -     12.00 -
------------------------------------------------------------------------
-Unknown or Not        -           -           -           -           -
-Reported              -         0 -         0 -         0 -         0 -
------------------------------------------------------------------------
-Ethnicity Category:   -           -           -           -           -
-Total of All Subjects -      6.00 -      6.00 -         0 -     12.00 -
------------------------------------------------------------------------
```

6 PROBLEM: COMBINING SUMMARY DATA WITH INDIVIDUAL RECORDS

Sometimes, summary descriptive statistics need to be calculated, and then associated back with an individual value in the database. For example, if you want to associate a percentile of a value with that particular record, the percentile needs to be computed, and then linked to the individual record. This can be done by creating an output dataset, and then using the `set` command in a `data step`.

6.1 SCENARIO

In this example, `PROC UNIVARIATE` is used to create an output dataset with one record, which contains the summary statistics of interest (percentiles) of the original variable `prop_scores`. To associate the output statistics with the original file, two `set` statements are used. Then, using basic `if - then` statements can be used to classify which quintile the original value came from.

6.2 CODE

```
data prop_scores;

     do i = 1 to 30;
           pred_prob = ranuni(345343);
           output;
     end;

     keep pred_prob;
run;


/*  find quintiles of scores  */

proc univariate data=prop_scores;
   var pred_prob;
   output out=Pctls pctlpts  = 20 40 60 80
                    pctlpre  = prop_
                    pctlname = pct20 pct40 pct60 pct80;
run;



/*  add quintiles to data set with scores  */

data prop_scores2;
```

```
     if _n_ = 1 then set pctls;
     set prop_scores;
run;


/*  create variable prop_quint that                */
/*  categorizes propensity scores into quintiles.  */


data prop_scores2;
     set prop_scores2;

     if missing(pred_prob) then prop_quint = .;

         if          . < pred_prob <= prop_pct20 then prop_quint = 1;
         if prop_pct20 < pred_prob <= prop_pct40 then prop_quint = 2;
         if prop_pct40 < pred_prob <= prop_pct60 then prop_quint = 3;
         if prop_pct60 < pred_prob <= prop_pct80 then prop_quint = 4;
         if prop_pct80 < pred_prob               then prop_quint = 5;
run;
```

6.3 OUTCOME

Each row in the `prop_scores2` database now has an indicator of which quintile the original value of `pred_prob` is in.

7 PROBLEM: ATTRIBUTES OF YOUR DATA ARE UNKNOWN

It may be the case that you are analyzing some data with which you are not totally familiar. It is sometimes helpful in coding to know the particular length of a variable, or whether a variable is character or numeric.

7.1 SCENARIO

You are working with a dataset supplied by someone else, and you would like to know the characteristics of the variables you are going to be using. The following code creates a dataset with each variable in the dataset and all the characteristics associated with each variable.

7.2 CODE

```
proc sql;
  create table AllVars as
    select *
      from dictionary.columns

  where libname = 'SASHELP' and memname = 'CLASS' ;

      ***** !!!!! note that the libname and memname above  must be all upper-case ;
quit;



proc print data=AllVars;
run;
```

7.3 OUTCOME

Variable attributes like: name, type, and length are saved in the new dataset `AllVars`. It is also possible to get information for **all** the variables in a particular library. You would just not include a `memname` value .

libname	memname	memtype	name	type	length	npos	nvar	nullable	format	informat	idxusage	sortedby	xtype	notnull	precision	scale	transcode
SASHELP	CLASS	DATA	Name	char	8	24	1					0	char	no	.	.	yes
SASHELP	CLASS	DATA	Sex	char	1	32	2					0	char	no	.	.	yes
SASHELP	CLASS	DATA	Age	num	8	0	3					0	num	no	.	.	yes
SASHELP	CLASS	DATA	Height	num	8	8	4					0	num	no	.	.	yes
SASHELP	CLASS	DATA	Weight	num	8	16	5					0	num	no	.	.	yes

8 PROBLEM: CODING PATTERNS OF VARIABLES INTO ONE VARIABLE

In some data collection situations, there are variables that are collected as "check all that apply". One situation is when race / ethnicity data is collected. Under the most recent Federal standards, the categories for race are:

1. American Indian / Alaska Native
2. Asian
3. Black or African American
4. Native Hawaiian or other Pacific Islander
5. White
6. More than one race

It is also possible to want to report whether the race is "Unknown" or "Not reported" . Generally, the data for race is collected as multiple variables that have to be coded into a single variable called "race". The general strategy presented here works for any type of data collection system that takes multiple variables and then aggregates the information into one variable.

8.1 SCENARIO

Several independent variables are used to code racial information for study participants. Race needs to be reported out as a single variable conforming to the latest Federal guidelines. Since we know that any case where more than one race category is checked will be counted as "More than one race," we can use some SAS functions that will allow us to use patterns and counts of things to create the new race variable. Specifically, we will use the `count()` and `cats()` functions. The `cats()` function concatenates the values of the variables you specify, and `count()` returns the count of the number of times the specified value occurs.

8.2 CODE

```
data demos;
input  id  ethnicity  raceai racea  raceaa racenh racew raceunk racenr ;
cards;
1  2 0 0 0 0 1 0 0
2  2 0 0 0 0 1 0 0
3  2 0 0 1 0 1 0 0
4  2 0 1 0 0 0 0 0
5  1 0 0 1 0 0 0 0
6  2 1 0 0 0 0 0 0
7  1 0 0 0 0 0 1 0
8  1 0 0 0 0 1 0 0
9  1 0 0 0 0 0 0 1
10 3 0 1 0 1 1 0 0
run;

data race; set demos;

*****  recode ethnicity to group together Unknown and Not Reported  ;

if ethnicity = 3 or ethnicity = 4 then rep_ethnicity = 3;
else rep_ethnicity = ethnicity;

format rc_race $char5. rc_unk $char2. ;


***** this "recodes" the race variables into one variable of length 5 made up of 0's and 1's  ;

rc_race = cats(raceai, racea, raceaa, racenh, racew);
rc_unk =  cats(raceunk, racenr);

***** this gives the count of the number of times a 1 occurs in:  rc_race and rc_unk  ;

c_rc_race = count(rc_race, "1") ;   *  counts the number of times a 1 occurs in rc_race ;
c_rc_unk  = count(rc_unk , "1") ;   *  counts the number of times a 1 occurs in rc_unk  ;


***** this gives the race category that matches the pattern  ;

if rc_race = "10000" then rep_race = 1;   *  i.e., american indian / AK native ! and nothing else
if rc_race = "01000" then rep_race = 2;   *  i.e., asian  ! and nothing else  ;
if rc_race = "00100" then rep_race = 3;   *  i.e., african american ! and nothing else ;
if rc_race = "00010" then rep_race = 4;   *  i.e., native hawaiian ! and nothing else  ;
if rc_race = "00001" then rep_race = 5;   *  i.e., white ! and nothing else  ;


*  if the count of 1's is greater than 1, the person is multi-racial  ;
if c_rc_race > 1 then rep_race = 6;       *  i.e., more than 1 race ;

*  if the count of 1's is greater than 0, the person's race is unknown or not reported ;
if c_rc_unk > 0 then rep_race = 7;        *  i.e., unknown or not reported ;

label  rep_race = "Racial Categories"
rep_ethnicity = "Ethnic Category"  ;

run;
```

```
proc format;

    value   frep_race       1 = "American Indian / AK Native"
                            2 = "Asian"
                            3 = "Black or African American"
                            4 = "Native Hawaiian or Other PI"
                            5 = "White"
                            6 = "More than one race"
                            7 = "Unknown or not reported"  ;

    value   frep_ethnicity  1 = "Hispanic or Latino"
                            2 = "Not Hispanic or Latino"
                            3 = "Unknown or Not reported"  ;

run;



proc freq data = race;
format rep_race frep_race. ;
tables rep_race ;
run ;
```

8.3 Outcome

The resulting variable `rep_race` is one variable that includes all the possible outcomes of the several *original* race variables.

The SAS System

The FREQ Procedure

Racial Categories

rep_race	Frequency	Percent	Cumulative Frequency	Cumulative Percent
American Indian / AK Native	1	10.00	1	10.00
Asian	1	10.00	2	20.00
Black or African American	1	10.00	3	30.00
White	3	30.00	6	60.00
More than one race	2	20.00	8	80.00
Unknown or not reported	2	20.00	10	100.00

9 PROBLEM: KEEPING ONLY THE OUTPUT YOU WANT

SAS produces lots of output for certain procedures, however, you may only be interested in a few parts of the output. SAS does offer a way to select which parts you want. You can use the ODS options to select the output that is desired. Most procedures in SAS produce tables that have particular names, and the output delivery system (ODS) can be used to pick which ones you want. The following example uses PROC LIFETEST.

9.1 SCENARIO

Using the ODS options, keep only the Quartiles, SurvDiff, and SurvivalPlot elements from the output.

9.2 CODE

```
data selection;
input weightloss $ surv_month   cens @@;
datalines;
No  1   1    No  11  1    Yes 15  1
No  38  0    No  6   0    Yes 13  0
No  9   0    No  9   1    Yes 14  1
No  9   1    No  4   1    Yes 71  0
No  13  1    No  20  1    Yes 37  0
No  20  1    No  44  1    Yes 17  1
No  8   0    No  81  0    Yes 32  0
No  4   1    No  74  0    Yes 17  1
No  27  1    No  13  1    Yes 13  1
No  62  1    Yes 6   1    Yes 27  1
No  12  1    Yes 39  1    Yes 8   1
No  32  1    Yes 100 1    Yes 63  1
No  17  1    Yes 30  0    Yes 22  1
No  15  1    Yes 1   1    Yes 5   1
No  15  1    Yes 2   1    Yes 62  0
No  8   0    Yes 2   0    Yes 9   1
No  22  1    Yes 6   1    Yes 15  1
No  7   1    Yes 21  1    Yes 5   1
No  16  1    Yes 20  0    Yes 11  1
No  11  0    Yes 5   1    Yes 21  1
;
run;

*****  selects from the ods output tables: Quartiles, SurvDiff ;
*****  selects frou the ods output plots: SurvivalPlot   ;
```

```
ods select Quartiles SurvDiff SurvivalPlot ;
proc lifetest data=selection;
time surv_month*cens(0);
strata weightloss/ diff=all test=logrank;
run;
ods select off;
```

If you don't know the name of the particular table[2], you can use the `ods trace on` option to get the names of all the output objects from a particular procedure printed in the log. For example:

```
ods trace on ;
proc lifetest data=selection;
time surv_month*cens(0);
strata weightloss/ diff=all test=logrank;
run;
ods trace off;
```

will show the ODS name for each object produced in the output.

9.3 OUTCOME

The output is exactly the same as normal, just only the selected objects are presented in the results.

[2] An internet search for "SAS ods table names proc __" will usually give you a page with a listing of the tables that are produced by the ODS system.

10 PROBLEM: KNOWING WHEN YOU DID YOUR ANALYSIS

Sometimes, it is difficult to remember or find any documentation of when an analysis was run. This is especially true when working in a team, and only having output from other team members. The following code adds footnotes to output to include the date and time of the analysis.

10.1 SCENARIO

When working on a research team, results are shared among team members. You want to add a note to the output so that everyone knows what parts were done by whom. Some small macros can be used to specify a user-specific footnote.

10.2 CODE

```
%LET USER = Johnny Awesome ;
%LET CURTIME=%SYSFUNC(TIME(),TIMEAMPM11.0);
%LET CURDATE=%SYSFUNC(TODAY(),WORDDATE.);
FOOTNOTE "THIS OUTPUT WAS CREATED AT &CURTIME ON &CURDATE BY &USER" ;

data selection;
input weightloss $ surv_month   cens;
datalines;
No  1   1    No  11  1    Yes 15  1
No  38  0    No  6   0    Yes 13  0
No  9   0    No  9   1    Yes 14  1
No  9   1    No  4   1    Yes 71  0
No  13  1    No  20  1    Yes 37  0
No  20  1    No  44  1    Yes 17  1
No  8   0    No  81  0    Yes 32  0
No  4   1    No  74  0    Yes 17  1
No  27  1    No  13  1    Yes 13  1
No  62  1    Yes 6   1    Yes 27  1
No  12  1    Yes 39  1    Yes 8   1
No  32  1    Yes 100 1    Yes 63  1
No  17  1    Yes 30  0    Yes 22  1
No  15  1    Yes 1   1    Yes 5   1
No  15  1    Yes 2   1    Yes 62  0
No  8   0    Yes 2   0    Yes 9   1
No  22  1    Yes 6   1    Yes 15  1
No  7   1    Yes 21  1    Yes 5   1
No  16  1    Yes 20  0    Yes 11  1
No  11  0    Yes 5   1    Yes 21  1
;  run;
```

```
*****  is there an association between weight loss status and censored cases?  ;

proc freq data = selection;
tables weightloss*cens / all relrisk;
run;
```

10.3 OUTCOME

The output will have the footnote with the date, time, and the specified user name.

```
                          The FREQ Procedure
...(snip)

              Statistics for Table of weightloss by cens

Statistic                     DF       Value      Prob
------------------------------------------------------
Chi-Square                     1      0.0243    0.8762
Likelihood Ratio Chi-Square    1      0.0243    0.8762
Continuity Adj. Chi-Square     1      0.0000    1.0000
Mantel-Haenszel Chi-Square     1      0.0239    0.8772
Phi Coefficient                       0.0201
Contingency Coefficient               0.0201
Cramer's V                            0.0201


             Fisher's Exact Test
----------------------------------
Cell (1,1) Frequency (F)          8
Left-sided Pr <= F           0.6729
Right-sided Pr >= F          0.5534

Table Probability (P)        0.2263
Two-sided Pr <= P            1.0000

THIS OUTPUT WAS CREATED AT 3:06:12 PM ON July 30, 2013 BY Johnny Awesome
```