Paper B2B-04

A few Useful tips when working with SAS®

Milorad Stojanovic, RTI International, North Carolina

ABSTRACT

Often working very hard to meet deadlines we stick to old proven solutions. But SAS is a live software and with each release / version we should learn new features which will apply efficiencies and make our working life easier. Improvements in PROC SORT, combining various old/new SAS functions, doing more in less number of steps (DATA or PROC) improves performance, readability and frequently maintenance of SAS programs. Paper would deal with matching data, checking duplicates, creating more compact Excel files, and validating data in output non SAS files.

KEYWORDS Merge, Sort, Excel, SortN, Compare

INTRODUCTION

Several years ago I spotted an article in information technology publication under the name "Working with Dull Knifes" (Lit. 1). At first glance I was surprised by the title - what would be connection between "dull knifes" and producing efficient and easy maintenance software. After I finished reading the article, I understood and was completely convinced by Clarke Ching's argument. Mr. Ching wrote "**Our productivity and sense of professionalism are dependent on the skill and experience we bring to the job and the tools we have**" and argued that just like the butcher needs to keep his knives sharp, IT professionals need to continually hone their skills and keep on top of changes within the software applications they use.

Five possible improvements/tips are described below. There are a large number of other new SAS functions but time and space is limited. Numerical examples and corresponding timing values would be available at presentation.

1. Merging two SAS data sets

While merging two SAS data sets may seems like an easy task - simply sort two data sets by same variables and do data step with MERGE statement – but this view makes several assumptions. First, that "We are living in an ideal world" – where our data files contain clean and perfectly well entered data from perfect documents or interviews done with no mistakes or wrong spellings. If this is true there is no need to read the rest of this section. But usually our working life is not so easy because we are dealing more or less with "dirty" data.

Let say that our task is to match two files received from different sources so we would be able to have more descriptive info about some event, person, etc.

Solution.
Sort both SAS data sets and merge it by set of 'by' variables. Yes, we will get some number of matches bigger or smaller depends on the level of quality of 'by' variables data. Based on what you were told from other side ('their data were checked and was in good shape') you expected more matches.

First improvement
Create for both data sets auxiliary variables to match length of smaller for each pair of string 'by' variables. Assign values to auxiliary variables (later AUX will be used) from corresponding original variables. Convert all string variables – AUX variables (in both SAS data sets) to small or capital letters. What is obvious to humans it is not for computer. Soon you would see number of matches is going up.

Second improvement
Remove suffix (like JR., SR, III, etc.) if exists as a part of family name at the both data sets in AUX variables only. Removing it in all corresponding auxiliary variables we are comparing "apples" with "apples". You would see slightly more matched cases.

Third improvement
Convert all non-English letters to the corresponding English letters. In US it is reality of presence of significant percentage of Latino population. Very often one source allows entering non-English letters (like Spanish or

Portuguese special letters) and other source of data doesn't allow it. You would see further improvement in number of matches.
Solution for it is:

Aux_Var = translate(Aux_Var,"eeeaaaaeiiiinooooouuuu",
"èéêáâãäëìíîïñòóôõöùúûü") ;

Fourth improvement
Find and remove famous invisible characters '0A0D' carriage return and/or feed new line. Again more matches would be achieved. By doing the following you hit two birds with one stone. You are keeping for names just letters of English alphabet and also you removed invisible characters. Note this **is not valid** for address variable(s). Why – an address could have numbers, special characters etc. Eventually you should convert letters to small but to keep all other characters as is.

Aux_Var = compress(Aux_Var, 'abcdefghijklmnopqrstuvwxyz', 'k') ;

Fifth improvement
Instead of doing merge by number of 'by' variables one composite 'by' variable should be created which includes values of all 'by' variables. This approach has few advantages. First you are using one variable only instead of several. Second we create 'clone' variables for each of 'by' variables but we don't use them later after assigning value to Composite variable. Third we would not be in situation to make mistake and to do all previously described transformations on original variables because we are creating one new composite variable. A new SAS function CATS provides the ability to concatenate multiple fields into a single index variable (as is an elegant replacement for "||").

Example: By variables are Name, Company, and Address.
Composite_Var = CATS(aux_Name, aux_Company, aux_Address);

Example of content Composite variable.
Composite_Var ="stojanovicmiloradnonamecomp313nostreetnotown99999"

All said above should be done in two preparatory data steps and after that just sort and merge these newly created SAS data sets. In the recent year's author found rising number of string variables with imbedded 'invisible' characters. Author doesn't have an answer why it is but be prepare to experience/handle it.

2. Checking duplicates in data set

Very often we experience a need for checking on duplicates in SAS data sets. Usually we should not have duplicates after certain point in our SAS program. There is more than one way to do this and here is the 'old' way and also a new way provided by SAS. This is not anything brand new but would be good to use more and more features provided by SAS than to do similar on our own.

When we are looking for duplicates we can do it either by way a. or by b.

Proc sort data = input_test out = out_test nodupkey ;
By var1 var2 var3 ;
Run ;

After that we should check the SAS LOG to see if still there are duplicates in INPUT_TEST SAS data set. If there are duplicates and we like to see/process them one we should execute one more SORT without NODUPKEY option, and data step to get duplicates. Simple but do we need to do that. In this example observations with var1, var2, and var3 values are the key variables in finding duplicate records.

We should use new feature in PROC SORT – DUPOUT.

*Proc sort data = input_test out = out_test DUPOUT = **duplicates** nodupkey ;*
By var1 var2 var3 ;
Run ;

We can use very small but useful macro **NumObs** (code is attached in the appendix) to determine the number of observations in **duplicates**. If number is 0 we were resolved all duplicates and we could proceed further. If not, we can take a second look at the content of **duplicates** SAS data set and apply proper actions.

3. Using new format of Excel files

SAS provides us with few easy ways to create / export SAS data set to Excel. It is already known fact. Now we have an opportunity to create Excel files in the new XML format by using option DBMS = XLSX in PROC Export. Author suggests using this new option as a must. You would find that newly created Excel file has **several times smaller physical size** that if the one created as XLS file. That is very important because now you are almost not limited to **65,535** rows (new max number of rows is **1,000,000**), less time to create excel file and less time to open file. If you are in situation to provide client with Excel file of 250,000 rows it means you should send to him **four** files (XLS format) instead one (XSLX format).

NOTE – If client still is using Excel 2002/2003 don't do this because he/she would not be able to open and use data. Another important condition is that SAS on production machine is the same version and same upgrade [example: SAS 9.3 (TS1M2)]. If you have for example upgrade TS1M2 and production machine Ts1M1 don't be surprised if perfectly correct excel file (from your prospective) could not be read by client if he/she has TS1M1 upgrade.

4. Checking data in output file

This is looking obvious but MUST be done each and every time we are delivering data to client (internal or external). Many times we are in situation to send data files to client (not mandatory in SAS data set format) and we are uneasy is everything is OK or even if file is readable. Under assumption we know what is software installed at client site we should do some checks before "press" the button for sending data to client. Please note this is critical step and after lot of effort if data are not in readable / usable form it could have serious consequences (loosing of client confidence, no bonus, or even worsened).

Let say we should deliver CSV file and we have on hand SAS data set ('image' copy) of what we need to deliver. One possible solution is the following.

a. Convert SAS data set to CSV file by using Proc Export or your preferred choice of SW (ODS etc.).
b. Convert back this CSV file back to SAS data set (new one) by using again Proc Import or your choice of SW.
c. Use Proc Compare to compare ORIGINAL (base) and DELIVERED (compare) SAS data sets with option absolute.
d. If listing of Proc Compare is short (no discrepancies found) you are ready to "press" send button.

Do it every time you are sending data to client. If you do it you would not be sorry later.

5. Sorting but not with Proc Sort

When we are talking about sorting we usually are talking about sorting observations in SAS data set in predefined order by given variable(s). Here we like to point out something different. Over the years author had from time to time (not very often) need to sort values of several variables (usually it was one dimension array in the same observation) in ascending or descending order. For that purpose was developed SAS code but somehow, after not in use for year or two it was forgotten and task (of developing code) was done again. Recently out of blue author found a neat SAS SortN call routine (for numeric variables). Officially it was introduced in SAS 9.3 but author tested it also under SAS 9.2 and it was working fine.
Please take a look first at author's code and later at SAS provided call routine.

```
%let totNum = 8 ;
array Num(&totNum.) Num1 Num2 Num3 Num4 Num5 Num6 Num7 Num8;
length aux 8 ;
aux = . ;
do j = 1 to (&totNum. - 1) ;
   do i = 1 to (&totNum. - 1) ;
      if Num(i) < Num(i+1) then ;
      else do ;
         aux = Num(i) ;
         Num(i) = Num(i+1) ;
```

```
            Num(i+1) = aux ;
        end ;
    end ;
end ;
drop aux i j;
```

Now same problem if you need to sort in ASCENDING order is solved in one line only with exception of %let statement.

```
%let totNum = 8 ;
array Num(&totNum.) Num1 Num2 Num3 Num4 Num5 Num6 Num7 Num8;
Call SortN(of Num(&totNum.)) ;
```

This reduces the number of lines from 15 to 3, and is more efficient resulting in a faster program. There is no question which code is easier for maintenance. Please note there is complementary SortC call routine for string variables. One additional remark – author's original SAS code is easy to be changed to DESCENDING order.

LIMITATIONS

Code was tested to run under SAS 9.2 and 9.3 under Windows XP. It was not tested on other platforms or other versions/releases of SAS.

CONCLUSION

- Keep in mind we are not living in ideal world – "dirty" data are around us.
- Incrementally improve your program(s).
- Be curious and do not stop learning.
- Hone your skills – how? In each SAS publication (manual) you should find chapter "What is new in SAS 9.x". Read it carefully and make notes what would be useful for you.
- Attend SAS user's group conferences from global to local. If it is not an option for you there are still thousand SAS papers available on the Internet.
-

ACKNOWLEDGEMENTS

The author would like to thank Peter Einaudi and Jean Lennon from the Education Studies Division (RTI) for all their help, comments and support in producing this paper.

REFERENCES

1. Clarke Ching, "Working with Dull Knifes" , Better Software, Volume 8, Issue 1
2. SAS® 9.3 Language Reference Concepts, Second Edition, 2012,SAS Institute, Cary, NC
3. SAS® 9.3 Functions and CALL Routines, 2011,SAS Institute, Cary, NC
4. SAS® 9.3 Statements Reference, 2011,SAS Institute, Cary, NC
5. Base SAS® 9.3 Utilities: Reference, 2011,SAS Institute, Cary, NC

DISCLAIMER

All opinions and suggestions stated in the paper on how to do processing of data do not necessarily reflect the opinions and suggestions of the Education Studies Division (RTI International) or RTI International as whole.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Milorad Stojanovic
Education Studies Division
RTI International
3040 Cornwallis Rd

RTP, NC, 27909
Work Phone : (919) 541-7376
E-mail: milorad@rti.org

TRADEMARK INFORMATION

APPENDIX

```
%macro numobs (dsn=,_count_=);
  %local nobs dsnid;
  %let nobs=.;

  %* Open the data set of interest;
  %let dsnid = %sysfunc(open(&dsn));

  %* If the open was successful get the;
  %* number of observations and CLOSE &dsn;
  %* Number of observations = 0 means data set exists but no observations in it ;
  %if &dsnid %then %do;
      %let nobs=%sysfunc(attrn(&dsnid,nlobs));
      %let rc  =%sysfunc(close(&dsnid));
  %end;
  %else %do;
      %let nobs=-1; %* value of -1 means data set does not exist or was corrupted ;
      %put Unable to open &dsn - %sysfunc(sysmsg());
  %end;

  %* Return the number of observations;
  &nobs
  %let &_count_=&nobs ;
%mend numobs;
```