PharmaSUG 2016 – Paper QT18

PROC SQL: To Create Macro Variables with Multiple Values and the Uses in Clinical Programming

Anish Kuriachen, inVentiv Health Clinical, NJ, USA

ABSTRACT

It is very easy to do programming with help of only data steps and procedures. But at times this might not be efficient. Especially when dealing/working with multiple datasets with same operations. For example we may need to copy multiple datasets from one location to other location. We could do this using multiple data steps but it will be very lengthy and we may miss some of the variables or datasets in the operations. In these cases we could use of macro variable with multiple values and PROC SQL. The intent of this paper is to present a method to handle the macro variables with multiple values and its usage to work or modify multiple datasets and multiple variables in a dataset. This logic could be used in clinical programming in various scenarios.

INTRODUCTION

In the world of SAS®, most programmers come quickly to the realization that any given task can be accomplished by in various ways. There is never a right way or a wrong way of doing things. But there is always a more or less efficient way of achieving results. Many people may like to use data steps, while others may prefer Proc SQL. Although opinions may vary on the “right” or “better” way to do things, the goal of this paper is not to argue whether or not my method of programming approach is superior to another method adopted by other programmers, but I am sure this will be a good solution may come handy in our day to day programming logic.

CREATING MACRO VARIABLE

In order to create a macro variable with multiple values we could use PROC SQL and SELECT statement shown

```
proc sql noprint;
select distinct (<<var>>) into : <<Macro Var>> separated by ‘<<any character>>’
from <<dataset name>>;
/*Storing the Variable Counts to serve as future loop/array size*/
select  distinct(count(<<var>>)) into:<<N>> from <<dataset name>>;
quit;
```

It is good to give “noprint” option in the PROC SQL because we could avoid displaying all the values in output window which are already stored in to the macro variables. When you create a macro variable with multiple values, always remember to create a macro variable with the counts of total values included in the macro variable with multiple values. After the PROC SQL call, the values can be selected using SELECT statement followed by DISTINCT option to choose only unique values and then the INTO statement which will write the values to the macro variable, followed by a “SEPARATED BY” statement which will allow you to place a separator between the multiple values and followed by FROM statement to specify the dataset name. The dataset name may include library name also like “work.dsname”. The Second “Select” is used to create the number of values stored in the macro variable, which will be used as the maximum value in loop statements or size of the array.

DEALING WITH THE LOOPING OPERATION INSIDE DATASET

When we do any kind of looping operations inside a data statement, it is better to use arrays by assigning the values from macro variable; you could use many variables to do the operations like renaming the variables, checking any missing values in any variables inside a dataset and etc. Sample code will be like below.

```
data <<dataset2>>;
    set <<original dataset>>;
    array <<arrayvar1>> {N}  (&<<macro var>>.);
    /*Sample statement : array bionms {&N} $50 _temporary_ (&bionm.) */
    Do i=1 to &N;
      <<Your code to manipulate variables and its values like check and drop
        missing values etc.>>
    end;
run;
```

DEALING WITH LOOPING OPERATION TO WORK WITH MULTIPLE DATASETS

Sometimes we need to concatenate multiple datasets in to a final dataset, this time below statements is very useful for managing the situation. Also same code may be used for sorting or adding new variables or finding frequencies etc.

```
%do i=1 %to <<&N>>;
%let <<dsname>>=%scan(<<&macrovar>>, <<&i>>);
<<
Now the dataset name is read to the <<dsname>> variable, so you can use this as the
reference and do all the necessary operation.
>>
%end;
```

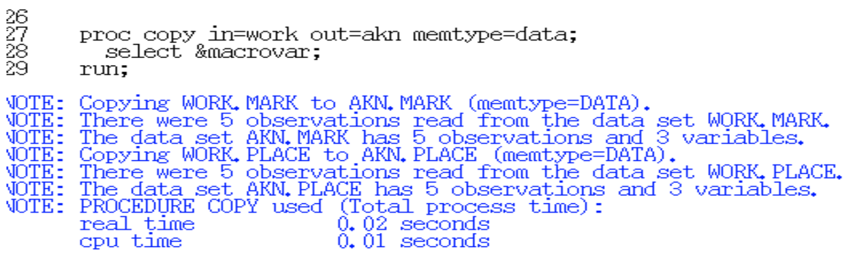
In the above code we can see a loop where the dataset name is being read into macro variable DSNAME and after that you could use whatever the steps you want to use for moving/modifying multiple datasets at same time. For example if you want to concatenate multiple datasets in a library then this method could be really helpful. You may also use same code for moving selected datasets from one location to another location.

EXAMPLE 1: COPYING DATASETS

```
data mark;
 id=1;name=”Milan”;mark=98;ouput;
 id=2;name=”Jillian”;mark=99;ouput;
 id=3;name=”Vivek”;mark=99;ouput;
 id=4;name=”Vinod”;mark=98;ouput;
 id=5;name=”Vipin”;mark=99;ouput;
run;
data place;
 id=1;name=”Milan”;place=”Paramus”;ouput;
 id=2;name=”Jillian”; place=”Newark”;ouput;
 id=3;name=” Vivek”;place=”Paramus”;ouput;
 id=4;name=” Vinod”; place=”Newark”;ouput;
 id=5;name=” Vipin”; place=”Newark”;ouput;
run;
proc sql noprint;
  select distinct (memname) into : macrovar separated by ‘ ’ from dictionary.tables
  where libname=’WORK’;
quit;

proc copy in=work out=akn memtype=data;
 select &macrovar;
run;
```

In figure 1 you could see the log for “Proc Copy”.



```

     proc copy in=work out=akn memtype=data;
       select &macrovar;
     run;

NOTE: Copying WORK.MARK to AKN.MARK (memtype=DATA).
NOTE: There were 5 observations read from the data set WORK.MARK.
NOTE: The data set AKN.MARK has 5 observations and 3 variables.
NOTE: Copying WORK.PLACE to AKN.PLACE (memtype=DATA).
NOTE: There were 5 observations read from the data set WORK.PLACE.
NOTE: The data set AKN.PLACE has 5 observations and 3 variables.
NOTE: PROCEDURE COPY used (Total process time):
      real time           0.02 seconds
      cpu time            0.01 seconds
```

Figure 1. Log for “Proc Copy”

You can use a do loop for copying the dataset using “Data step” so that you could filter the datasets name. For example if you don’t want to copy all the datasets from a library then you could specify the dataset names in a “if” statement like “if <<dataset name>> not in (‘<dataset name 1> ‘<dataset name 2>’…)”. Main usage for this method is copying data from base area to snapshot areas.

EXAMPLE 2: DOING OPERATIONS IN MULTIPLE DATASETS

Sometime we face to do many unique steps for different datasets in a library like sorting, printing or writing the result to an external file. Below code is showing how to generate Excel files from different datasets.

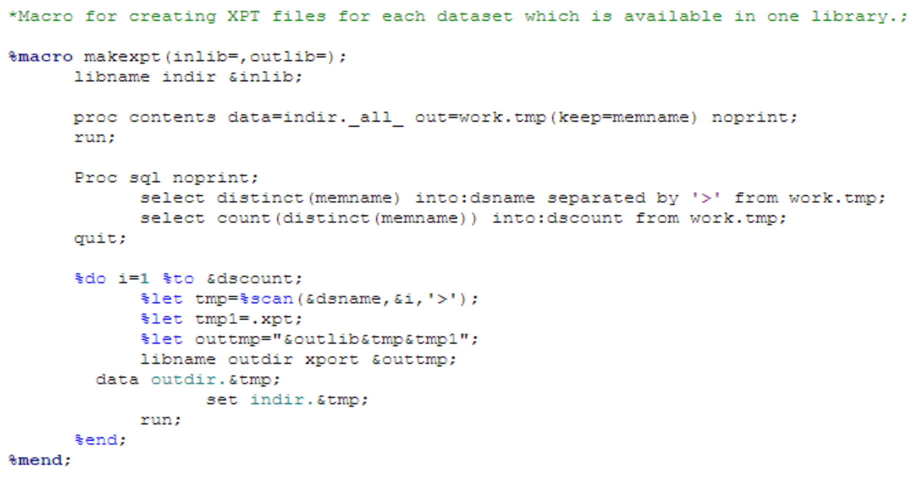
```
%macro test();
   %do i=%to &n;
     %let dsname=%scan(&macrovar),&i);
     ODS HTML FILE=”<<path>>/&dsname._excel.xls”
              HEADTEXT=”<style>
                        Td {mso-number-format:\@};
                        BR {mso-data-placement:same-cell}
                        </style>”;
     Proc print data=&dsname noobs;
     Run;

     ODS HTML CLOSE;
   %end;
%mend;
```

You could do many number of operations inside the loop, also you could add conditions like ignore some datasets with name starts with particular letters, total number of observations. Usually you will encounter some errors while doing operations in a null dataset, hence it is always good to check if dataset is null or not before passing it to the macro for certain operations.

EXAMPLE 3: CREATING XPT FILES FOR MULTIPLE DATASETS

Another use of this method is to convert datasets to XPT files for transferring datasets from one version to another version of SAS®, also which could be very useful to create xpt files for Submission. Figure 2 shows the code for generating XPT files.



```
*Macro for creating XPT files for each dataset which is available in one library.;

%macro makexpt(inlib=,outlib=);
      libname indir &inlib;

      proc contents data=indir._all_ out=work.tmp(keep=memname) noprint;
      run;

      Proc sql noprint;
            select distinct(memname) into:dsname separated by '>' from work.tmp;
            select count(distinct(memname)) into:dscount from work.tmp;
      quit;

      %do i=1 %to &dscount;
            %let tmp=%scan(&dsname,&i,'>');
            %let tmp1=.xpt;
            %let outtmp="&outlib&tmp&tmp1";
            libname outdir xport &outtmp;
        data outdir.&tmp;
                  set indir.&tmp;
            run;
      %end;
%mend;
```

Figure 2. Macro for generating XPT files.

Proc contents is used to get all the datasets names to a dataset called TMP and then from this TMP dataset, the datasets names are copied to the macro variable DSNAME. This is macro program which I created to generate XPT files from the datasets which is created in SAS 6 and copied it to use in SAS 8. So if you are working in a project to migrate the datasets between 2 versions, this method of coding will be really helpful. Also this method is useful for creating XPT files and creates define files in submission macros.

CONCLUSION

The intent of this paper was to present a method to handling the macro variables with multiple values and the usage in clinical programming. Above examples is using minimum number of datasets or variable, but this method will be really useful when you will be working with big number of datasets. The examples given can be expanded much further but it is beyond the scope of this introductory paper.

REFERENCES

http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/viewer.htm#a001360983.htm

http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000543554.htm

ACKNOWLEDGMENTS

I would like to thank inVentiv Health Clinical, my Associate Director Mathew A Bryant, mangers Yong Zhao and Rajesh Malani respectively for encouraging and supporting me to write and present the paper.

CONTACT INFORMATION

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

Name	: Anish Kuriachen
Enterprise	: inVentiv Health Clinical
Address	: 111 North 17th Street
City, State ZIP	: Prospect Park, NJ-07508
Work Phone	: (732) 567-5975
E-mail	: anish.kuriachen@inventivhealth.com