PhUSE 2010

Paper CC02

Convert dataset variables into macro variables: a new approach with the CALL SET routine and the SCL FETCH function

Patrice AGATOR, Sanofi-Aventis, Chilly-Mazarin, France

ABSTRACT

Combine the CALL SET routine to convert dataset variables into macro variables and the SCL FETCH function to automatically produce multiple and customized reports easier than ever.

With an example using laboratory data, I will illustrate this new approach and will try to explain the use of these non-friendly SAS statements.

At the end, people who attend this presentation will be able to implement this new powerful technique if they remember only a few lines of SAS code.

INTRODUCTION

Even after more than 15 years working as a SAS programmer in pharmaceutical industry, I still find pleasure in learning new SAS tricks and tips. Once you are aware of a new SAS programming technique, the way you do things in the future may never be the same.

I am going to share my recent experience with you and I will show you how you can easily convert SAS dataset variables into macro variables and create multiple and customized reports using a new approach.

DISCOVERY OF A NEW SAS CODE

Some years ago, I discovered a new SAS code developed for the production of patient profiles (see the SAS code below). The result of this SAS code is multiple reports repeated for each subject identifier in the input dataset work.ADDM.

```
%MACRO patient_profile;

    %let dsid=%sysfunc(open(work.ADDM,I));
    %if &dsid gt 0 %then %do;
        %syscall set(dsid);
        %do %while (%sysfunc(fetch(&dsid)) eq 0);

            title1 "Unique Subject IdentifIer = %trim(&usubjid)";

            title2 "Demography";
            proc print data=ADDM noobs;
              var age race dmbegdt dmenddt;
              where usubjid eq "%trim(&usubjid)";
            run;

            title2 "Laboratory Data";
            proc print data=ADLB noobs;
              var lbtestcd visitnum lbdt lbstresn;
              where usubjid eq "%trim(&usubjid)";
            run;

       %end;
    %end;
    %let rc=%sysfunc(close(&dsid));
%MEND patient_profile;

%patient_profile;
```

You can easily see that this SAS code creates two outputs for any given patient (usubjid), one for demographics information and the other for laboratory information. But the main question remains: "Where are the different values for macro variable *usubjid* coming from"? Then you look for SAS documentation and you find interesting things about the CALL SET routine (Links SAS data set variables to DATA step or macro variables that have the same name and data type) and the FETCH function (Reads the next non-deleted observation from a SAS data set into the Data Set Data Vector (DDV)).

Attached, are the links for the description of both SAS statements:

http://support.sas.com/documentation/cdl/en/lrdict/62618/HTML/default/a000127856.htm

http://support.sas.com/documentation/cdl/en/lrdict/62618/HTML/default/a000202928.htm

Once you discovered this interesting technique, you will be interested in using it widely. A case study on laboratory data is described below.

THIS NEW SAS TECHNIQUE IN ACTION

From the CDISC standard analysis dataset containing the laboratory data I want to create a set of summary tables and box plots, each represents a laboratory test across visits. With this new technique it will be easier than in the past.

SUMMARIZE YOUR LABORATORY DATA

First I need to create a dataset work._pilot_ which contains for each parameter all the information needed for the reporting (name of the parameter, label of the parameter, category and sub-category of the parameter, minimum and maximum values for both x and y axis on the graph).

```
proc sql;
  create table _pilot_ as
  select distinct lbcat, propcase(lbscat) as LBSCAT, lbtestcd, lbtest
         ,min(visitnum) as lower_xaxis
         ,max(visitnum) as upper_xaxis
         ,min(lbstresn) as min_lbstresn
         ,max(lbstresn) as max_lbstresn
         ,case lbtestcd
               when "HB" then 10
               when "NA" then 5
               when "K" then 0.5
               else 1
          end as step label="Step definition for graphs"
         ,floor(min(lbstresn/calculated step))*(calculated step) as lower_yaxis
         ,ceil(max(lbstresn/calculated step))*(calculated step) as upper_yaxis
  from ADLB
  where lbstresn ne .
  group by lbtestcd
  ;
  quit;
```

The FLOOR function takes one numeric argument and returns the largest integer which is less than or equal to the argument. The CEIL function also takes one numeric argument and returns the smallest integer which is greater than or equal to the argument. The PROPCASE function converts an argument to proper case (i.e. uppercase the first character of each word).

Table _pilot_ has one line for each laboratory test (3 in the example).

	LBCAT	LBSCAT	LBTESTCD	LBTEST	lower_xaxis	upper_xaxis	min_lbstresn	max_lbstresn	step	lower_yaxis	upper_yaxis
1	BIOCHEMISTRY	Electrolytes	K	POTASSIUM	1	4	3.2	5.7	0.5	3	6
2	BIOCHEMISTRY	Electrolytes	NA	SODIUM	1	4	113	148	5	110	150
3	HEMATOLOGY	Red Blood Cells And Platelets	HB	HEMOGLOBIN	1	4	94	180	10	90	180

CONVERT DATASET VARIABLES INTO MACRO VARIABLES AND CREATE MULTIPLE REPORTS

The second part of the program creates the macro variables and also creates the summary reports for each laboratory parameter.

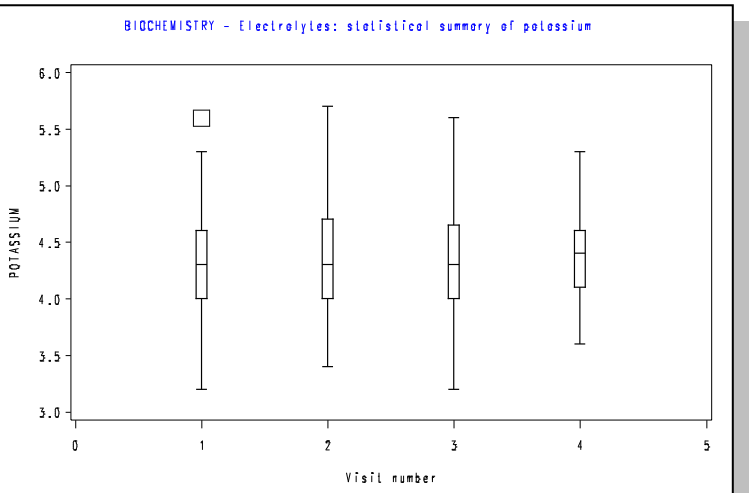
After opening the work._pilot_ dataset with the SCL OPEN function, the CALL SET routine (%SYSCALL) and the SCL FETCH function will convert the variables in work._pilot_ dataset into macro variables and assign values to these macro variables. These values would be the values of the variables in work._pilot_ and they would change with each loop according to the value in work._pilot_ dataset. CALL SET and SCL FETCH are then followed by reporting procedures where macro variables can be used as usual. In the following example, macro variables will be used for the selection of the parameter, the title description and the definition of the lower and upper x and y axis.

```
%MACRO lab_report;

    * -- Open the _pilot_ dataset which contains each laboratory test (I corresponds
         to Input mode);
    %let dsid=%sysfunc(open(_pilot_,I));

    %if &dsid gt 0 %then %do;

        * -- Invokes the SAS CALL SET routine (it links SAS data set variables
             to macro variables) (i.e. creates &lbcat, &lbscat, &lbtestcd,...);
        %syscall set(dsid);

        * -- Loop the _pilot_ data set for each observation (laboratory test);
        %do %while (%sysfunc(fetch(&dsid)) eq 0);

            title1 "%trim(&lbcat) - %trim(&lbscat): summary of %lowcase(&lbtest)";

            proc tabulate data=ADLB formchar=" ----------" noseps format=6.2;
              class visitnum visit;
              var lbstresn;
              tables visitnum="Visitnum",lbstresn=" "*(n*f=6. mean std min max) /
                     box="&lbtest";
              where lbtestcd eq "%trim(&lbtestcd)";
            run;

            title1 c=blue h=0.8 f=TimesRoman
                   "%trim(&lbcat) - %trim(&lbscat): statistical summary of %lowcase(&lbtest)";

            symbol1 value=square interpol=boxt10 h=2 w=2.0 c=black;

            axis1 label=(H=0.8 A=0 R=0 F="TimesRoman" "Visit number")
                  value=(H=0.8 F="TimesRoman")
                  order=(%eval(&lower_xaxis-1) to %eval(&upper_xaxis+1) by 1 ) minor=none;

            axis2 label=(H=0.8 A=90 R=0 F="TimesRoman" "%trim(&lbtest)")
                  order=(&lower_yaxis to &upper_yaxis by &step)
                  value=(H=0.8 F="TimesRoman") minor=none;

            proc gplot data=ADLB;
              plot lbstresn * visitnum / haxis=axis1 vaxis=axis2;
              where lbtestcd eq "%trim(&lbtestcd)";
            run;

        * -- END OF DO WHILE;
        %end;

    %* -- END OF DSID > 0;
    %end;

    %let rc=%sysfunc(close(&dsid));

%MEND lab_report;

%lab_report;
```

The result of the SAS code is a table and a graph produced for each laboratory parameters where titles and axis ranges are automatically updated at each loop.

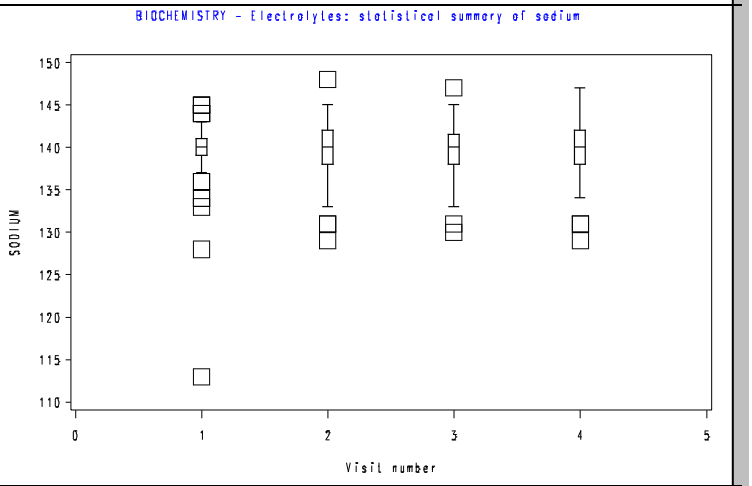
BIOCHEMISTRY - Electrolytes: statistical summary of potassium

POTASSIUM	N	Mean	Std	Min	Max
Visitnum					
1	79	4.31	0.47	3.20	5.60
2	79	4.31	0.48	3.40	5.70
3	72	4.33	0.53	3.20	5.60
4	67	4.35	0.40	3.60	5.30



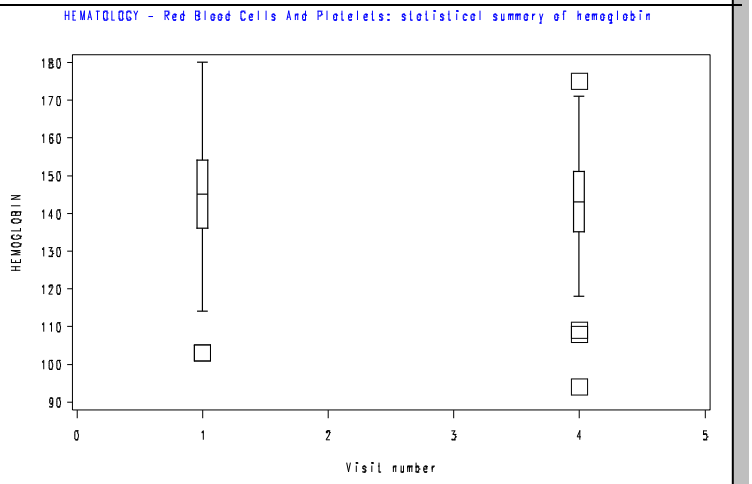
BIOCHEMISTRY - Electrolytes: statistical summary of sodium

SODIUM	N	Mean	Std	Min	Max
Visitnum					
1	78	139.60	4.16	113.00	145.00
2	79	139.86	3.18	129.00	148.00
3	72	139.82	2.96	130.00	147.00
4	67	139.51	3.46	129.00	147.00



HEMATOLOGY - Red Blood Cells And Platelets: statistical summary of hemoglobin

HEMOGLOBIN	N	Mean	Std	Min	Max
Visitnum					
1	77	143.60	15.83	103.00	180.00
4	65	142.34	15.24	94.00	175.00



JUST REMEMBER SHORT SAS CODE

As you have seen with the previous example, it will become easy to obtain customized reports with a very short SAS code from a summary dataset. This SAS code can be applied for a large type of data organized the same way in clinical studies (ECG, Vital Signs, Questionnaire,…).

Once you create a summary dataset (work._pilot_), you just have to remember a few lines of SAS code and you have got your macro variables ready to use. The only modification you need to do in the code below is to replace the name of the input dataset (work._pilot_) and to create your SAS code for the reporting.

```
%MACRO resume;

    %let dsid=%sysfunc(open(work._pilot_,I));
    %if &dsid gt 0 %then %do;
        %syscall set(dsid);
        %do %while (%sysfunc(fetch(&dsid)) eq 0);

            * -- include your SAS code --;

       %end;
    %end;
    %let rc=%sysfunc(close(&dsid));
%MEND resume;
```

CONCLUSION

This new way to create macro variables is easy and powerful. Just try it.

RECOMMENDED READING

SAS Global Forum 2007: Changing Data Set Variables into Macro Variables (William C. Murphy)
This paper compares this SAS technique with other methods to get macro variables.

CONTACT INFORMATION

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

Patrice AGATOR
Sanofi-Aventis
1, avenue Pierre Brossolette
91385 CHILLY-MAZARIN Cedex
Phone: +33 (0) 1 60 49 40 01
Email: patrice.agator@sanofi-aventis.com