Processing Large Lists of Parameters and Variables with SAS® Arrays and Macro Language

Eugene Tsykalov, GlaxoSmithKline, King of Prussia, PA

ABSTRACT

Use of SAS arrays and Macro language is demonstrated to automatically handle large lists of variables and parameters in normalized (vertical) datasets without using the names of the variables (parameters) again and again. Practical examples include manual transpose of normalized datasets and validation of Laboratory parameters in a clinical trial data.

INTRODUCTION

Normalized datasets are a common way to store large number of parameter values such as Laboratory data in a Clinical Trial. Usually one variable in a normalized dataset contains names of the parameters, while other variables have values and units for these parameters.

Processing large numbers of variables and parameters in a normalized dataset could be a challenging task. For example, checking correctness of units for particular parameters, validating the presence of a set of parameters, and crosschecking of variables of different types may require long chains of “if- then”, “where” and “select” statements.

This paper presents an elegant solution to automate this process by creating macro variables containing lists of processed variables and/or parameters of any length.

PROCESSING LISTS OF PARAMETERS

Subsetting normalized dataset to a list of parameters

Suppose our simplified Laboratory dataset LAB has the following variables:

- PATIENT patient number
- TESTDATE date of lab test
- LPARM contains names of parameters
- VALUEN has numeric values of parameters
- UNIT has a text value of parameters units

Normalized dataset with Laboratory data could easily have dozens of parameters.

To check only few of them, let’s create a macro variable which contains the list of parameters and use it to subset the lab dataset:

```
%let parm_list = granulocytes neutrophils
                 eosinophils  basophils lymphocytes
                 monocytes  wbc ;

data lab1 ;
  set lab  ;

   i = 1 ;
  do while ( scan("&parm_list", i, ' ') ^= ' ' ) ;
   if ( upcase(lparm) = scan(upcase("&parm_list"), i, '
') ) then output ;
    i + 1 ;
  end ;
 drop i ;

run ;
```

In this code, the first SCAN() function was used to sequentially go in a DO WHILE loop through a list of processed parameters separated by spaces. The second SCAN() function is used to return a name of a parameter from a list to compare it with a dataset parameter name for subsetting.

Checking Units of Parameters

Each of the parameters could have different valid units. With seven parameters and five possible units for each parameter, the regular code to check each parameter for each unit may be way too long.

The following code accomplishes this task using a macro variable with a list of valid units. It checks if each parameter has a unit from this list. If a parameter has an incorrect unit it will be output with a discrepancy reason.

```
%let unit_list = 10^9/L 10^3/mcL 10^3/mm^3  L/L % ;

data invalid_units ;
  set lab1 ;
  length reason $20 ;

  i = 1 ;
  do while ( scan("&unit_list", i, ' ') ^= ' ' ) ;
  if indexw( unit, scan("&unit_list", i, ' ') ) then delete ;
  i + 1 ;
  end ;

  if unit = ' ' then reason = "No unit" ;
               else reason = "Not valid unit" ;
  drop i ;
```

```
run ;
```

Now we use the SCAN() function to go through a list of valid units in a DO WHILE loop. The INDEXW() function checks if the value of variable UNIT is in the units list. Observations with incorrect units are routed to an output dataset.

PROCESSING LISTS OF VARIABLES

Manual Transpose of a Normalized Dataset Converting Parameters to Variables

If we need to crosscheck parameters with each other in a normalized dataset, we have to have all compared parameters as variables values in the same observation. We can use powerful PROC TRANSPOSE to denormalize the dataset. Yet, if data is incomplete, for example not all parameters are present for all test dates, PROC TRANSPOSE could produce misleading results. In such a situation the old clean simple manual transpose would come to rescue.

Using SAS arrays and previous techniques we can elegantly transpose a normalized lab dataset:

```
proc sort data = lab1 ;
  by patient testdate ;
run ;

data lab2 ;
  set lab1 ;

  retain &parm_list ;
  by patient testdate ;

  array parms{*} &parm_list ;

  /* Initialise transposed values to missing */
  if first.testdate then
  do ;
     do i = 1 to dim(parms) ;
       parms{i} = . ;
     end ;
  end ;

  /* Transpose parameters values to variables
values */
  i = 1 ;
  do while ( scan("&parm_list", i, ' ') ^= ' ' ) ;
   if ( upcase(lparm) = scan(upcase("&parm_list"), i, '
') ) then
    do ;
      parms{i} = valuen ;
    end ;
    i + 1 ;
  end ;
  if last.testdate then  output ;

run ;
```

In this code we first use SAS arrays to create variables with names from a list of parameters and initialize those variable to missing values, as we will be using RETAIN statement to collect all parameter values in the same observation.

In the next step we use the SCAN() function to go through a list of parameters and assign created variables values to corresponding parameters: parms{i} = valuen ;

Calculating a Number of Parameters in a List

In previous tasks we did not need to know how many parameters or units to process. In some other processing tasks the number of parameters in a list could be used. To calculate it we can use the following code:

```
data _null_ ;
  set lab1 (obs = 1) ;

  array parms{*} &parm_list ;

    n1 = dim(parms) ;
    n1_c = put(n1, 3.) ;
    call symput( 'n_parm', left(trim(n1_c)) ) ;

run ;
```

The statement “array parms{*}” automatically calculates the number of elements to create. This number is then extracted with the function dim() and assigned to a macro variable n_parm to further use in the program.

Crosschecking Variables

Now that we have parameters values from a normalized dataset converted into variables values we can do some crosschecks on these values.

For example, in our Lab dataset, let’s do the following data checks:

- if a count of White Blood Cells (WBC) is missing then all other parameters should be missing (as they are components of WBC), otherwise create a discrepancy
- if a count of White Blood Cells (WBC) is not missing then all other parameters should be present (as they are components of WBC), otherwise create a discrepancy

Here is the code for these tasks:

```
data wbc_miss ;
  set lab2 ;

  length reason $20 ;
  array parms{*} &parm_list ;
  array diffr{*} diffr1-diffr&n_parm ;

  do i=1 to &n_parm ;
    diffr{i} = parms{i} ;
  end ;

/* If wbc is missing and at least one of other */
/* parameters is not missing                   */

  if (wbc = .) then
  do ;
    if ( n(of diffr2-diffr&n_parm) > 0 ) then
    do ;
      reason = "WBC miss/Other nmiss" ;
      output ;
    end ;
  end ;

/* If wbc is not missing and at least one of other */
/* parameters is missing                           */

  if wbc ^= . then
  do ;
    if ( nmiss(of diffr2-diffr&n_parm) > 0 ) then
    do ;
      reason = "WBC nmiss/Other miss" ;
      output ;
    end ;
  end ;

run ;
```

In order to automatically process a list of variables with different names we need first to create a duplicate list of variables diffr{} with a numeric suffix:

```
  array parms{*} &parm_list ;
  array diffr{*} diffr1-diffr&n_parm ;

  do i=1 to &n_parm ;
    diffr{i} = parms{i} ;
  end ;
```

For this task we will need the macro variable n_parm created before, which has a number of variables in the list.

Then we use functions n() and nmiss() to check the number of missing variables values:

```
    if ( n(of diffr2-diffr&n_parm) > 0 ) then  …
    if ( nmiss(of diffr2-diffr&n_parm) > 0 ) then …
```

and create a reason for discrepancy.

CONCLUSION

The techniques using SAS arrays and Macro language described in this paper allow us to automate the processing of large lists of parameters and variables without repeatedly using names of variables and parameters from a list.

These techniques could be useful in processing and data checking of large normalized datasets such as datasets with Laboratory data from Clinical Trials. They also could be used as building blocks wherever a processing of large lists of variables and/or parameters is required.

CONTACT INFORMATION

Eugene Tsykalov
GlaxoSmithKline
2301 Renaissance Blvd
RN0420, P.O.Box 61540
King of Prussia, PA 19406-2772
Tel: (610)787-3855 (w)
Fax: (610)787-7074
e-mail: Eugene.2.tsykalov@gsk.com