Processing Large Lists of Parameters and Variables
With Macro Techniques

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 subsetting and manual transpose of normalized datasets and cross-checking 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:

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>patient ID number</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESTDATE</td>
<td>date of lab test</td>
</tr>
<tr>
<td>LPARM</td>
<td>name of parameter</td>
</tr>
<tr>
<td>VALUEN</td>
<td>numeric value of parameter</td>
</tr>
<tr>
<td>UNIT</td>
<td>text value of parameter units</td>
</tr>
</tbody>
</table>

The structure of such a normalized Lab dataset may look like this:

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>TESTDATE</th>
<th>LPARM</th>
<th>VALUEN</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>01/01/03</td>
<td>lymphocytes</td>
<td>100</td>
<td>10^3/mcL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monocytes</td>
<td>130</td>
<td>10^3/mm^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wbc</td>
<td>330</td>
<td>10^9/L</td>
</tr>
<tr>
<td></td>
<td>02/11/03</td>
<td>hemoglobin</td>
<td>90</td>
<td>g/L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lymphocytes</td>
<td>105</td>
<td>10^3/mcL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wbc</td>
<td>300</td>
<td>10^9/L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rbc</td>
<td>3.3</td>
<td>10^12/L</td>
</tr>
<tr>
<td>002</td>
<td>01/05/03</td>
<td>monocytes</td>
<td>110</td>
<td>10^3/mm^3</td>
</tr>
<tr>
<td></td>
<td>02/15/03</td>
<td>wbc</td>
<td>420</td>
<td>10^9/L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hemoglobin</td>
<td>95</td>
<td>g/L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>glucose</td>
<td>7.3</td>
<td>mM/L</td>
</tr>
</tbody>
</table>

In Clinical Trial normalized dataset with Laboratory data could easily have dozens of parameters. To process only few of them, let’s create a macro variable that contains the list of parameters of interest 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.

SUBSETTING DATASET WITH QUOTED LIST OF PARAMETERS

In previous example we used a list of parameters separated by spaces. Same task could be coded even more elegantly with a quoted list of parameters separated by commas.

First we create the regular list of parameters:

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

Now let's convert it into a quoted list, like 'granulocytes','neutrophils','eosinophils', etc.:

data _null_;  
a  = &parm_list ;
t1 = compbl(a) ;
t2 = '"' || trim(left(t1)) || '"' ;
t3 = tranwd(trim(left(t2)), " ", "','" ) ;
call symput('parm_q', t3) ;
run ;

In variable t1 the function COMPBL() is used to remove any extra spaces in original macro variable list. It makes the code more robust, for example, it will work if you have more than one space between any two parameters in the list. Then we add the first and last quotes to the original list (variable t2). Next, in variable t3 function TRANWD() is used to replace spaces between parameters with ',', to make all parameters quoted and separated by commas.

The final macro variable could also be created in one step:

data _null_;  
a  = &parm_list ;
t = "" || tranwd(trim(left(compbl(a))), " ", "','" ) || "" ;
call symput('parm_q', t) ;
run ;

Now we can use the quoted list of parameters &parm_q with operator IN() for various purposes:

For subsetting dataset:

data lab1 ;
  set lab ;
  if lparm in(&parm_q) ;
run ;

For use with proc print of the dataset:

proc print lab ;
  where lparm in(&parm_q) ;
run ;
CHECKING UNITS OF PARAMETERS

Each of the laboratory parameters could have different valid units. For example, 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.

With quoted list of units the code would look like this:

Create quoted list of correct units:

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

data null ;
a = &unit_list ;
t = || tranwrd( trim( left(compbl(a)) ), " ", " "," ") || " " ;
call symput ( 'unit_q', t ) ;
run ;
```

Using quoted list of parameters &parm_q and quoted list of units &unit_q, we can reduce subsetting of original lab dataset and checking units to one line of code (!):

```
data invalid_units ;
  set lab ;
  length reason $20 ;
  if lparm in(&parm_q) and unit in(&unit_q) then delete ;
  if unit = ' ' then reason = "No unit" ;
  else reason = "Not valid unit" ;
run ;
```

While use of quoted lists of parameters/units in these examples is more elegant, the approach with DO WHILE loop and SCAN() function is more universal and will work in many other cases where use of IN() operator may be limited.
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:

```sas
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:

```sas
array parms{*} &parm_list ;
```

and initialize those variables 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 values for created variables to the values of corresponding parameters:

```sas
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:

```sas
data _null_ ;
    array parms{*} &parm_list ;
    nl = dim(parms) ;
    nl_c = put(nl, 3.) ;
    call symput( 'n_parm', left(trim(nl_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:

```plaintext
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 diffr1-diffr&n_parm) > 0 ) then
        do ;
          reason = "WBC miss/Other nmiss" ;
          output ;
        end ;
    end ;
  end ;

  /* If wbc is not missing and at least one of other parameters is missing */
  if wbc ^= . then
    do ;
      if ( nmmiss(of diffr1-diffr&n_parm) > 0 ) then
        do ;
          reason = "WBC nmiss/Other miss" ;
          output ;
        end ;
    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:

```plaintext
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:

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

and create a reason for discrepancy.
The example above illustrates the use of SAS arrays. The same task of cross-checking the values of the parameters could be achieved without arrays, if to notice that macro variable &parm_list contains the list of variables whose names are the names of parameters (after transpose above):

```sas
data wbc_miss;
  set lab2;
  length reason $30;

  /* If wbc is missing and at least one of other parameters is not missing */
  if (wbc = .) then do;
    if ( n(of &parm_list) > 0 ) then do;
      reason = "WBC miss/Other nonmiss";
      output;
    end;
  end;

  /* If wbc is not missing and at least one of other parameters is missing */
  if wbc ^= . then do;
    if ( nmiss(of &parm_list) > 0 ) then
      do;
        reason = "WBC nmiss/Other miss";
        output;
      end;
  end;
run;
```

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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.