Paper 103-2008

The Application of General List Processing Macro in Combination Therapy

Zhongwen Huang, Walgreens, Deerfield, IL
William Layton, Walgreens, Deerfield, IL

ABSTRACT

List is one of the common data structures that we use in our daily work. There are many solutions in SAS to deal with this problem. Most of them will require a series of macro functions to build a general list processing target. After that, we still need to write a special purpose macro for each different mission. This increases the code complexity. In this paper, we will present a general list processing macro widely used in our work. This list macro has the following characteristics: 1. Simplicity. Simple is the best and the macro only contains several statements. 2. Flexibility. The macro is powerful and can solve many problems. 3. Versatility. Only one macro function will be needed in different situations. No other sub-macro procedures need to be created. Finally we show the application of this macro in a Combination drug therapy program.

Keyword: List macro; General list processing macro; Combination therapy

INTRODUCTION

Combination therapy is a method of treating disease by using one drug to replace the simultaneous use of a variety of drugs. It can increase patients' compliance and persistence, while generally decreasing the drug prescription cost. The kernel part of this program is to identify the possible Combination therapy chances, e.g. two types of therapeutic class drugs which appear within a certain number of days (we assume it is 90 days here). Because each therapeutic class is a unique number and there are various classes, the total therapeutic classes' process can be a list. The purpose of this paper is to show the application of a simple general list processing macro in the Combination therapy program. Certainly, the list processing macro can also be applied to other applications.

EXAMPLE 1: USE GENERAL LIST PROCESSING MACRO TO REPLACE VARIABLE NAMES

Here is an example of Rx claims data in a Combination therapy application.

Patient	Service Date	GCN_ SEQ_ NO	g68	g70	g85	g95	g257	g306	g307	g323	g354
000000000000000005	19JAN2007	23382	0	0	0	0	0	0	0	0	0
000000000000000005	05SEP2006	18368	0	0	0	0	0	0	0	0	0
000000000000000005	09JAN2007	29968	0	0	1	0	0	0	0	0	0
000000000000000006	05DEC2006	29968	0	0	0	0	0	1	0	0	0

For each patient, there might be multiple drug claims. On each claim, g68, g70 and g85 etc stand for whether the drug in the claim belongs to a certain therapeutic class, 0 means NO while 1 means YES, which have the possibility to be in a Combination therapy. Other variables also exist in the dataset.

First step is to reduce dataset size by only keeping the variables of interest. The therapeutic class numbers that can be combined for a Combination therapy are in variable list1 and variable list2 of a SAS file, TherapeuticClassList.

So we can generate a list with following codes:

```
* Step 1: Get TherapeuticClass List;
Proc sql;
select distinct TherapeuticClass into :List separated by " "
        from
        (
           select list1 as TherapeuticClass from TherapeuticClassList
          union
          select list2 as TherapeuticClass from TherapeuticClassList
         )
        order by TherapeuticClass
         ;
```

```
Quit;
%put &List;
```

We get a therapeutic class list that can be used for Combination therapy:
68 85 95 257 306 307 323 354 377 520 577 591 660 844 876 1300

Then we need to keep the project interested variables. Three methods can be used to deal with this.

A: EXPLICITLY LIST ALL VARIABLES WHEN THE NUMBER OF LIST ITEMS IS NOT TOO LONG.

```
Data Out;
   Set claim (keep=patient ServiceDate GCN_SEQ_NO
                    g68 g85 g95 g257 g306 g307 g323 g354 g377
                    g520 g577 g591 g660 g844 g876 g1300
             );
Run;
```

Two disadvantages of this method are: 1. when list items change, we need to modify the code and retype all affected variables in this part, which makes the code difficult to maintain, especially when the program becomes complicated. This programming style is prone to errors. 2. It makes the typing a burden when the list is long.

B: USE MACRO FUNCTION

We can use the macro function to replace variable names.

```
%MACRO glist (mList=);
          %local j g;
          %let j= 1;
          %let g= %scan(&mList,&j);
          %do %while (&g ^=);
            g&g
            %let j = %eval(&j+1);
            %let g = %scan(&mList,&j);
          %end;
%MEND glist;
Data Out;
     Set claim(keep=patient ServiceDate GCN_SEQ_NO %glist(mList=&List));
Run;
```

The advantage of this method is we need not type in the variable name for each item in the list. All names can be generated automatically once there is the list. We can also write another macro if a different type of list name is required. The disadvantage of this method is we have to write different list macro functions when the variable name changes. It makes the work tedious. For example, to create New_68_SEQ, New_85_SEQ, New_95_SEQ etc will require another macro with similar structure.

C: GENERAL LIST PROCESSING MACRO

Due to the disadvantages, we now modify the above macro and make it a general list processing macro.

```
%MACRO GeneralList(mList=&List, Param=%nrstr(g&Item));
           %local j Item;
          %let j=1;
          %let Item=%scan(&mList,&j);
          %do %while (&Item ^=);
               %unquote(&param)
               %let j=%eval(&j+1);
               %let Item=%scan(&mList,&j);
           %end;
%MEND GeneralList;
```

Then the code can be written as follows:

```
Data Out;
```

```
        Set claim(keep=patient ServiceDate GCN_SEQ_NO    %GeneralList(mList=&List, Param=%nrstr(g&Item)) );
Run;
```

In this modified macro, we use a keyword parameter to pass the different types of transforms of the list item and %nrstr() & %unquote() macro functions in the macro definition to resolve the macro variable later.

The %nrstr() macro function prevents g&Item from being resolved during macro compilation. Thus, the value g&Item is what actually gets passed as &param. The %unquote() macro function then allows g&Item to be resolved during macro execution.

This macro can be used for other types of list transform. If we need to create New_68_SEQ, New_85_SEQ, New_95_SEQ etc, simply use %GeneralList (mList=&List, Param=%nrstr(New_&Item._SEQ)).

This macro is simple, yet flexible. It embeds the list item number in the parameter directly and saves the time writing a special purpose macro every time. We can use this macro consistently in the data step and PROC SQL. It sacrifices the encapsulation because the list item macro variable, &Item, appears in the keyword macro parameter. We must include them in our code and have to use the same name all the time.

EXAMPLE 2: USE GENERAL LIST PROCESSING MACRO TO REPLACE THE EXPRESSIONS OR SINGLE STATEMENTS

In addition to the replacement of list variable names, we can also use the macro to replace expressions and some statements.

A: EXPRESSIONS

For the dataset in example 1, we want to filter the claims to those where at least one of the interesting therapeutic classes is true.

We use the same macro for implementation.

```
Step 2: Focus in eligible therapeutic class only;
Data Out1;
        Set Claim (keep=patient ServiceDate GCN_SEQ_NO %GeneralList);
             Where %GeneralList (param=%nrstr(g&Item not =0 or)) 0=1;
Run;
```

After the macro resolve, the codes become:

```
Data Out;
Set claim(keep=patient ServiceDate GCN_SEQ_NO %GeneralList(mList=&List, Param=%nrstr(g&Item)) );
      WHERE (g68 not = 0) or (g85 not = 0) or (g95 not = 0) or (g257 not = 0) or (g306 not = 0) or (g307 not = 0) or
                   (g323 not = 0) or (g354 not = 0) or (g377 not = 0) or (g520 not = 0) or (g577 not = 0) or (g591 not = 0)
                   or (g660 not = 0) or (g844 not = 0) or (g876 not = 0) or (g1300 not = 0) or (0=1);
Run;
```

Attention:

1. Because the default value of the keyword parameter of “mlist” in generalist macro is &List and the “param” is %nrstr(g&Item), we need not fill in any value for the variable reduction part – the KEEP dataset option.
2. We must add 0=1 in the ‘where’ statement in SAS data step in order to make the syntax correct. There will be a “dangling OR operator” at the end of the macro generated text. For the “OR” operator, the added condition should always be false. Similarly, if we use the “AND” operator, we can add 1=1. For the “dangling AND operator”, the added condition should always be true.

B: SINGLE STATEMENT IN SQL

After we filter our claims, we need to find the paired claims that qualify for a Combination therapy. This can be done by an inner join with its own table.

```
Data out1;
```

```
        Set Out1;
         N = _n_;
Run;
Step 3: must be in 3 months with different therapeutic class and in the same patient;
Proc sql;
        create table Out2 as
                  select A.N, B.N as B_N, A.patient, A.ServiceDate as A_ServiceDate, B.ServiceDate, A.GCN_SEQ_NO
                           as A_GCN_SEQ_NO, B.GCN_SEQ_NO %GeneralList(param=%nrstr( ,sum(A.g&Item,B.g&Item)
                           as g&Item ))
        from   Out1 as A, Out1 as B
        where  A.patient = B.patient and
                   A.GCN_SEQ_NO ^= B.GCN_SEQ_NO and
                   A.ServiceDate - B.ServiceDate <= 90 and
                   A.ServiceDate - B.ServiceDate >= 0
        order  by A.N;
Quit;
```

After resolution, it becomes:

```
Proc sql;
        create table Out2 as
             select A.N, A.patient_therapeutic_seq, A.ServiceDate as A_ServiceDate, B.ServiceDate,  A.GCN_SEQ_NO
             as A_GCN_SEQ_NO, B.GCN_SEQ_NO, sum(A.g68,B.g68) as g68,
             sum(A.g85,B.g85) as g85, sum(A.g95,B.g95) as g95,    sum(A.g257,B.g257) as g257, sum(A.g306,B.g306)
             as g306, sum(A.g307,B.g307) as g307, sum(A.g323,B.g323) as g323, sum(A.g354,B.g354) as g354,
             sum(A.g377,B.g377) as g377, sum(A.g520,B.g520) as g520, sum(A.g577,B.g577) as g577,
             sum(A.g591,B.g591) as g591, sum(A.g660,B.g660) as g660, sum(A.g844,B.g844) as g844,
             sum(A.g876,B.g876) as g876, sum(A.g1300,B.g1300) as g1300
        from    Out1 as A, Out1 as B
        where  A.patient_therapeutic_seq = B.patient_therapeutic_seq and
                   A.GCN_SEQ_NO ^= B.GCN_SEQ_NO and
                   A.ServiceDate - B.ServiceDate <= 90 and
                   A.ServiceDate - B.ServiceDate >= 0
        order  by A.N;
Quit;
```

By using the general list processing macro function, we implement this target with the same macro. After getting the output table, we can use some predefined rules to decide whether there are claims for a specific patient qualified for a Combination therapy.

EXAMPLE 3: USE GENERAL LIST PROCESSING MACRO TO FINISH SOME COMPLICATED ITEM PROCESS WITHOUT WRITING OTHER SPECIFIC MACROS.

For the same example in Combination therapy, we need to calculate the average daily cost for each therapeutic class, which may be used to approximate the program's saving later. Certainly, we can use a specific purpose macro to solve the problem, but the general list processing macro can also be used to handle this simply.

```
Proc datasets nolist;
        Delete cost;
Quit;
%GeneralList(param=%nrstr(
                                          Proc sql;
                                                  Create table tmp as
                                                  select &Item as ID,
                                                            sum(PAID_AMT)/sum(Days_Supply) as Cost
                                                  from out1
                                                  where g&Item =1 and paid_amt>=0
                                                       ;
```

```
                    Quit;
                     Proc append base=Cost data=tmp force;run;
                     ) );
```

Here we first remove the destination output file, cost, by using PROC DATASETS. No errors will be triggered even if the file does not exist. Then we put the statement into the general list processing macro directly. It will go through all items in the list and process them one by one until the end of the list.

CONCLUSION

We have shown the application of the general list processing macro in the Combination therapy program. Although the macro is simple, it is powerful and versatile. This macro is a very useful tool in our daily work and saves us lots of time. We believe it will serve others as well

REFERENCES

SAS Institute Inc. SAS OnlineDoc@ 9.1.3 Cary NC. SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Zhongwen Huang
Health Outcomes Statistical Analyst
Walgreens Health Services
1415 Lake Cook Road MS #L444, Deerfield, IL 60015
Phone: 847-964-6429 Fax: 847-964-6917
E-mail: Zhongwen.huang@walgreens.com
Or
William Layton
Health Outcomes Programmer
Walgreens Health Services
1415 Lake Cook Road MS #L444, Deerfield, IL 60015
Phone: 847-964-8206 Fax: 847-964-6917
E-mail: William.Layton@walgreens.com