Update A Two-Dimensional Matrix Using The Macro Facility

Jeff F. Sun, Blue Cross Blue Shield of North Carolina, Durham, North Carolina

Abstract

One of the most significant features of macro facility within the SAS System is its ability to make internal and dynamic updates possible. This paper will introduce a utility macro which can be used to update a two-dimentional matrix dynamically and internally. The macro code demonstrates how to use some very important macro functions and macro statements in the SAS macro language and related techniques to solve the complicated real world problems.

1. Introduction

The potential pay-back for time and efforts invested in well-designed, modular generic utility macros can be very large in creating very sophisticated self-modifying programs for gaining efficiency and accuracy, especially for the routine file maintenance. When one maintains a large macro-based information system, usually there are many macro variables needed to be updated each time period, for example, each month. In order to update the information accurately and efficiently, utility macros are definitely helpful during file maintenance. With this kind of utility macros, macro variables in the system can be dynamically and internally updated during program compilation.

This short paper consists of four sections. The first section, closing here, provides a brief introduction to the need for file maintenance. Section 2 describes the requirements from a real world problem. The next section details the mathematical derivation. The macro source code is discussed in the last section.

2. Description of Requirements

Here is the description of requirements from a real world problem in an insurance company. A macro-based information system is maintained monthly. There are three macro variables (i.e. LAGL1, LAGL2, and LAAGL3) used to assign initial values for three arrays in several programs. These values, sometimes called lag values, are employed to indicate how old each claim (i.e. each observation) is and to make appropriate mathematical adjustment for each claim later on. For a certain reporting purpose, 18 month data are considered enough to have complete claims. When the 18^{th} month is reached, it is thought that the claim with 18 month lag is complete. Beyond 18 months, all values will be assigned as 18 to indicate the completion of those claims. For example one maintains a system in April 1997, which is indicated by k_0 (i.e. month). The most current claim data files are available for March 1997. The macro variables are defined as Figure 1. Using a matrix notation to represent these expressions in Figure 1 gives a *3x12* matrix shown in Figure 2.

In order to use the matrix notation to solve this problem, the above matrix can be rearranged to obtain a generalized *3x12* matrix for the lag values (see Figure 4).

					k_0								
		Jan	*Feb*	*Mar*	**Apr**	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
1997	*LAGL1* =	2	1	0	**0**	0	0	0	0	0	0	0	0
1996	*LAGL2* =	14	13	12	11	10	9	8	7	6	5	4	3
1995	*LAGL3* =	18	18	18	18	18	18	18	18	18	17	16	15

Figure 1. Three Initialized Values for Array LAG1, LAG2, and LAG3

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1997	2	1	0	0	0	0	0	0	0	0	0	0
1996	14	13	12	11	10	9	8	7	6	5	4	3
1995	18	18	18	18	18	18	18	18	18	17	16	15

Figure 2. A 3x12 Matrix for Macro Variables

	Dec	Nov	Oct	Sep	Aug	Jul	Jun	May	Apr	Mar	Feb	Jan
1997	0	0	0	0	0	0	0	0	0	0	1	2
1996	3	4	5	6	7	8	9	10	11	12	13	14
1995	15	16	17	18	18	18	18	18	18	18	18	18

Figure 3. A Modified 3x12 Matrix

	Dec	Nov	Oct	Sep	Aug	Jul	Jun	May	Apr	Mar	Feb	Jan
*year*1	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	$C_{1,4}$	$C_{1,5}$	$C_{1,6}$	$C_{1,7}$	$C_{1,8}$	$C_{1,9}$	$C_{1,10}$	$C_{1,11}$	$C_{1,12}$
*year*2	$C_{2,1}$	$C_{2,2}$	$C_{2,3}$	$C_{2,4}$	$C_{2,5}$	$C_{2,6}$	$C_{2,7}$	$C_{2,8}$	$C_{2,9}$	$C_{2,10}$	$C_{2,11}$	$C_{2,12}$
*year*3	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	$C_{3,4}$	$C_{3,5}$	$C_{3,6}$	$C_{3,7}$	$C_{3,8}$	$C_{3,9}$	$C_{3,10}$	$C_{3,11}$	$C_{3,12}$

Figure 4. A General Expression for This Problem

The major role this utility macro plays is that given a value in year *1* the utility macro %UTILITY() can update all values in this matrix $[C_{i,j}]$. In other words, the goal is to create a dynamic utility macro and generate information dynamically for the system.

3. Mathematical Derivation

[**Proposition**]

Let k_0 be the initial value (month) retrieved from the current system. The element of the lag matrix $[C_{i,j}]$ for the system can be defined as

$$C_{i,j}= \begin{cases} 0, & if\ 12*(i-1)+j \le k_0+1 \\ 18, & if\ 12*(i-1)+j \ge k_0+19 \\ 12*(i-1)+j-k_0-1, & otherwise \end{cases}$$

where the year index *i=1,2,3*; and the month index *j=1, 2,..., 12*.

[**Proof**] In order to prove the above expression, it is easy for us to use an one-dimensional transformation matrix, which is an *1x36* matrix $[T_k]$ (the index *k=1,2, ... , 36*).

$[T_1, T_2, ..., T_{35}, T_{36}] = [C_{1,1}, C_{1,2}, ... , C_{1,12}, C_{2,1}, C_{2,2}, ..., C_{2,12}, C_{3,1}, C_{3,2}, C_{3,3} , ..., C_{3,12}]$

The relation between the year index *i*, the month index *j*, and index *k* can be defined as:

$$k = 12*(i-1)+ j,$$

where i=1,2,3; and j=1,2,3, ..., 12.

Since the values of these lags are incremented by one each month, the system date for the month is the best choice for the initial value. If initial value (i.e. current month) from the system is k_0, which is the month when the system is run, the data used is for the month k_0+1.

For the most current month data, 0 month lag is assigned for $[T_k]$ when the index $k=k_0+1$.

If $k \le k_0+1$, $T_k = 0$

For the index $k=k_0+2$, $T_k=1$.

When $k=k_0+3$, $T_k=2$, ... ,

When $k=k_0+18$, $T_k=17$,

When $k=k_0+19$, $T_k=18$,

When $k \ge k_0+20$, $T_k=18$, ...

By induction, we have:

if $k \le k_0+1$, then $T_k = 0$, (***1***)

if $k \ge k_0+19$, then $T_k=18$, (***2***)

otherwise, $T_k= k-k_0-1$, (***3***)

Substituting *k* with *i* and *j* in above (1), (2), and (3), the following expressions for $[C_{i,j}]$ can be obtained:

if $12*(i-1)+j \le k_0+1$, then $C_{i,j} = 0$, (***1'***)

if $12*(i-1)+j \ge k_0+19$, then $C_{i,j} = 18$, (***2'***)

otherwise, $C_{i,j} = 12*(i-1)+j- k_0-1$ (***3'***)

Writing these expressions (1'), (2'), and (3') in the following form completes the proof.

$$C_{i,j} = \begin{cases} 0, & if\ 12*(i-1)+j \le k_0+1 \\ 18, & if\ 12*(i-1)+j \ge k_0+19 \\ 12*(i-1)+j-k_0-1, & otherwise \end{cases}$$

where $i=1,2,3$; and $j=1, 2,..., 12$.

4. Utility Macro %UTILITY()

a. SAS Macro Source Code

Figure 5 gives a SAS macro source code for this utility. The automatic macro variable SYSDATE is used to accomplish the dynamic and internal updates (**Sun, 1997**).

```
%MACRO UTILITY(NEWDATE=&SYSDATE);
  %LOCAL MNLIST MONTH0 MINDEX LAG1 LAG2 LAG3;
  %GLOBAL LAGL1 LAGL2 LAGL3;
  %LET MONTH0=%SUBSTR(&NEWDATE,3,3);
  %LET MON0=1;
  %LET MNLIST=JAN DEC NOV OCT SEP AUG JUL JUN
                MAY APR MAR FEB JAN;
  %DO %UNTIL (%SCAN(&MNLIST,&MON0) = &MONTH0);
    %LET MON0 = %EVAL(&MON0+1);
  %END;
  %LET MINDEX=%EVAL(&MON0-1);
  %DO I=1 %TO 3;
    %DO J=1 %TO 12;
      %IF %EVAL(12*(&I-1)+&J) LE %EVAL(&MINDEX+1) %THEN
        %DO;
          %LET C&I&J=0;
        %END;
      %ELSE %IF %EVAL(12*(&I-1)+&J) GE %EVAL(&MINDEX+19)
%THEN
        %DO;
          %LET C&I&J=18;
        %END;
      %ELSE
        %DO;
          %LET C&I&J=%EVAL(12*(&I-1)+(&J-&MINDEX-1));
        %END;
    %END;
  %END;
  %DO P=1 %TO 3;
    %LET LAGG1 = &&C&P.12%STR( )&&C&P.11%STR( )
                 &&C&P.10%STR( )&&C&P.9;
    %LET LAGG2 = &&C&P.8%STR( )&&C&P.7%STR( )
                 &&C&P.6%STR( )&&C&P.5;
    %LET LAGG3 = &&C&P.4%STR( )&&C&P.3%STR( )
                 &&C&P.2%STR( )&&C&P.1;
    %LET LAG&P = &LAGG1.%STR( )&LAGG2.%STR( )&LAGG3;
  %END;
  %IF &MINDEX NE 12 %THEN
    %DO;
      %LET LAGL1=&LAG1;
      %LET LAGL2=&LAG2;
      %LET LAGL3=&LAG3;
    %END;
  %ELSE
    %DO;
      %LET LAGL1=&LAG2;
      %LET LAGL2=&LAG3;
    %END;
%MEND UTILITY;
```

Figure 5. Macro Code for the Utility Macro %UTILITY()

b. Invocation

This macro utility only has one keyword parameter *NEWDATA* as follows.

%UTILITY(NEWDATE=&SYSDATE)

The default value is highlighted that *%UTILITY()* will assume if you omit it. For the information system monthly run, it is not necessary to assign any value to this parameter because this macro utility can retrieve information from the hosting system. When in an unusual case, for instance, re-running the information system in a different month, one has to set up a value for this parameter.

%UTILITY(NEWDATE=03FEB97)

This utility macro makes it possible to perform dynamic and internal system update based on the information retrieved from the hosting platform.

Reference

Jeff F. Sun, (1997) "Usage of SYSDATE for Dynamic and Internal Updates during File Maintenance Run in the Macro-Based Information System", in *Proceedings of the Fifth Annual Southeast SAS Users Group Conference*, Jacksonville, Florida, pp.45-54.

Trademarks

Any questions or comments regarding the paper may be directed to the author:

Jeff F. Sun
Health Services Information And Analysis
Blue Cross Blue Shield of North Carolina
ET600, 3rd Floor
Durham, NC 27705
Phone: (919)-765-7854
E-mail: jeff_sun@scp11.bcbsnc.com