

A Simple Time Series Macro

Scott Hanson, SVP Risk Management, Bank of America, Calabasas, CA

ABSTRACT

One desirable aim within the financial industry is to understand customer behavior over time. Despite vast interest, I have seen that the goal frequently is unmet. Clearly, an array (apologies) of methodologies abounds for the analyst. One simple approach involves a SAS® macro (NB. the code is written in SAS® 9.2 for the PC). The macro generates a time series file and features options such as merging external data, maintaining the time series, and automatic labeling of attributes. Finally, I provide a practical example, especially suited for the beginning programmer, involving arrays and do-loop processing.

INTRODUCTION

Theoretically, a time series file can overcome the paradox of harnessing large quantitative data against the need to discern discrete behavior. This phenomenon tends to occur in organizations with large customer data bases, but not exclusively so.

Consequently, the time series macro below is designed to overcome these inherent analytical challenges by discrete category (e.g., customer or account number, patient id, test case).

TIME SERIES MACRO (CONCEPTUAL VERSION)

It will be important to recognize that the macro is in essence a rudimentary data step. In abstracted form,

```
DATA TIMESERIES;
    MERGE A1 (RENAME=(VAR1=VAR1_[N1] VAR2=VAR2_[N1]... (1)
           A2 RENAME=(VAR1=VAR1_[N2] VAR2=VAR2_[N2]...
           A3 RENAME=(VAR1=VAR1_[N3] VAR2=VAR2_[N3]...
           ...
           A[N] RENAME=(VAR1=VAR1_[N] VAR2=VAR2_[N]
    BY ID; (2)
RUN;
```

- (1) Read in a series of snapshot or periodic data. SAS data sets A1, A2, A3, etc., are sequenced: monthly, weekly, daily are typical sequential categories. Variables are renamed, which will facilitate array declarations after the time series file is created.
- (2) The BY variable *ID* is a unique identifier, patient or customer id, loan number, for example. The variable name *ID* is arbitrary. The data sets presuppose a sort order, as analysis will be dependent on this unique, variable identifier.

Note that prepared SAS data consisting of snapshot information is requisite.

TIME SERIES MACRO (ACTUAL CODE)

Let's review the actual time series macro step-by-step (the entire program is available in the Appendix). The code begins with a few declarations:

```
LIBNAME S '\\calsas01\Share\sasdata'; (1)
%LET OBS = 100; /* Test Number, MAX */ (2)
%LET BYVAR = ID; (3)
%LET ORIGVARS = &BYVAR ORIGINATION_DATE BIRTHDATE; (4)
%LET KEEPVARS = BALANCE INCOME AGE CREDITLIMIT FICOSCORE; (5)
```

%LET WHERE = ; (6)

- (1) Default location of input data and output time series file.
- (2) Allows for testing (otherwise enter *MAX*).
- (3) The unique identifier. The input, snapshot data need to be in sort order by this variable.
- (4) Constant, static variables (Birth date, Origination date, Name, SSN, for example).
- (5) These variables can change over time (Balance, Age, Income, and Credit Limit, for example).
- (6) Optional where clause that will be inserted for each data set that is read in the MERGE statement.

Only variables assigned to the macrovariable **KEEPVARS** will be renamed. Variables assigned to macrovariable **ORIGVARS** will be included in the time series but only by original name.

The main macro, **ARRAYTS**, consists of six lines of code and three macro calls, **INARRAYFILES**, **SUBS**, and **LABEL** in a single data step:

```
%MACRO ARRAYTS (FDAY, END) ;  
DATA S.ARRAYTS; (1)  
    MERGE %INARRAYFILES (&FDAY., &END.) (2)  
    ;  
    BY &BYVAR.;  
    IF %SUBS (&FDAY., &END.) (3)  
    ;  
    %LABEL (&FDAY., &END.) (4)  
    ;  
RUN;  
%MEND;
```

- (1) Default location of snapshot, input data and time series file.
- (2) The two parameters of macro **INARRAYFILES** are initial and ending date of the time series.
- (3) This is an option that writes out a series of variables produced by dataset option IN=. Determines whether or not the unique identifier must be present in each snapshot data set in order to be output.
- (4) Appends a date stamp to the label of arrayed variables. For each variable, the macro writes out "LABEL &CVAR.&I. = "&CVAR.&I. AS OF &DT.;" Hypothetically, a variable BALANCE32 may not be easily discerned, but its label reads "BALANCE32 AS OF 200907"

Macro **ARRAYTS** is invoked by passing two SAS dates, the beginning and ending dates of the time series. For example

%ARRAYTS('01JAN2004'D, '01JUL2009'D)

The listing of macro calls offers flexibility. Now let's discuss the macro calls individually.

INARRAYFILES, the first macro call, reads in static, snapshot data:

```
%MACRO INARRAYFILES (FIRST, LAST) ;  
    %GLOBAL DT;  
  
    %LET INTRV = %EVAL (%SYSFUNC (INTCK (MONTH, &FIRST., &LAST.)) + 1) ; (1)  
    %DO I = 1 %TO &INTRV. ; (2)
```

```

%LET DD = %EVAL(%SYSFUNC(INTNX(MONTH,&FIRST., &I.) - 1);
%LET YY = %SYSFUNC(YEAR(&DD.));
%LET MMM = %SYSFUNC(PUTN(%SYSFUNC(MONTH(&DD.)),Z2.));
%LET DT = &YY.&MMM.; (3)
S.AF&DT.(OBS=&OBS. IN=IN&I. (4)
    &WHERE.
    KEEP=&ORIGVARS. &KEEPVARS.
    RENAME=(%RENAME(&I.,&KEEPVARS.)) (5)
%END;
%MEND;

```

One preliminary remark, macros are mere text generators (insofar as the author's limited understanding!). Macro **INARRAYFILES** writes out the names of the input files following the *MERGE* keyword in the data step. The default naming convention of the input files is a date-stamp, *YYYYMM*. A different naming convention can be specified, hence the assignment of separate macrovariables, *DD*, *YY*, and *MMM*.

- (1) The INTCK function performs subtraction of two SAS dates. In this example, the value returned is the interval, in months, between the last date of the time series and the initial date. The z2. numeric format and PUTN function perform a leading zero conversion.
- (2) Defines macrovariable I, and increments.
- (3) Appends a date stamp to the input files.
- (4) The data set IN= option relates to the macro *SUB*, which will determine which observations will be output to the time series.
- (5) Macro *RENAME* passes two parameters, the increment and entire list of variables declared in *KEEPVARS*.

Now let's review macro **RENAME**:

```

%MACRO RENAME(G,VARLIST);
%LET VARCNT = 1;
%DO %UNTIL(&VARCNT EQ 0);
%LET CVAR = %SCAN(%STR(&VARLIST),&VARCNT); (1)
%IF &CVAR EQ %STR() %THEN %LET VARCNT = 0;
%ELSE %DO;
%LET VARCNT = %EVAL(&VARCNT + 1);
&CVAR=&CVAR.&G (2)
%END;
%END;
%MEND;

```

- (1) Within the %DO statement, the key statement is the macrovariable declaration of *CVAR*. The *SCAN* function searches a text string (&VARLIST.) for the *n*th word and returns its value. The *n*th word is &VARLIST. The default word delimiters apply, of which 'blank' is relevant. The variables identified in the macrovariable *KEEPVARS* are passed through the positional parameter, *VARLIST*.
- (2) The string inserted in the "RENAME=()".
- (3) The associated increment will be invaluable for array declaration and do loop processing in the SAS program discussed later.

Macro **LABEL** uses identical logic in macros **INARRAYFILES** and **RENAME**, while generating these label elements: a variable name, an index *I*, and a date stamp.

```

%MACRO LABEL(FIRST, LAST);
%LET VARCNT = 1;
%DO %UNTIL(&VARCNT EQ 0);
%LET CVAR = %SCAN(%STR(&KEEPVARS.),&VARCNT);

```

```

%IF &CVAR EQ %STR() %THEN %LET VARCNT = 0;
%ELSE %DO;
    %LET VARCNT = %EVAL(&VARCNT + 1);
    %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
    %DO I = 1 %TO &INTRV.;
        %LET DD = %EVAL(%SYSFUNC(INTNX(MONTH,&FIRST., &I.) - 1);
        %LET YY = %SYSFUNC(YEAR(&DD.));
        %LET MMM = %SYSFUNC(PUTN(%SYSFUNC(MONTH(&DD.)),Z2.));
        %LET DT = &YY.&MMM.;
        LABEL &CVAR.&I. = "&CVAR.&I. AS OF &DT."; (1)
    %END;
%END;
%END;
%MEND;

```

(1) Multiple label statements are generated and will be inserted within the data step.

The macro **SUBS** generates a concatenated string of IN= [OR|AND]. The choice of “AND” or “OR” determines whether the time series file contains by variable values present in every data set (i.e. reflected as a series of “AND” conditions), or a by variable value is present in at least one input data set. The default is union, that is, a series of *OR* conditions.

```

%MACRO SUBS(FIRST, LAST);
    %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
    %DO I = 1 %TO &INTRV. - 1;
        IN&I. OR (1)
    %END;
    IN&INTRV.
%MEND;

```

(1) Default is *OR*, which, in effect, keeps all by variable values. *AND*, the other option, keeps only by variable values that are present in every input data set.

MACRO TESTFILE: A SAMPLE EXERCISE

The macro **TESTFILE** illustrates how the time series can be used for analysis at the discrete level (e.g. customer identification number).

```

%LET BEGLOOP = 1; /* 1 = JUN2006 */ (1)
%LET ENDLOOP = 35; /* 35 = APR2009 */ (1)
OPTIONS OBS=MAX;

%MACRO TESTFILE;
DATA ARRAYTSFLAG_F;
    LENGTH DLQMBAFORCLTYPE_F $3;
    MERGE ARRAYTS (IN=AA)
        S.JUN2006_MAR2007_F1_24MO(IN=BB); (2)
    BY ID;
    IF AA AND BB;
    ARRAY MOB(*) MOB&BEGLOOP.-MOB&ENDLOOP.; (3)
    ARRAY DLQ(*) DLQ&BEGLOOP.-DLQ&ENDLOOP.; (3)
    ARRAY PRINBAL(*) PRINBAL&BEGLOOP.-PRINBAL&ENDLOOP.; (3)

    CONTINUE = 1;
    DO LOOP=BEGLOOP TO &ENDLOOP; (4)
        IF MOB(LOOP) = 0 THEN DO;
            PRINBAL_1 = PRINBAL(LOOP);
            END;
        IF CONTINUE THEN DO; (5)
            IF DLQ(LOOP) IN ('F1') THEN DO;
                MOB_F = MOB(LOOP) + 1;
                PRINBAL_F = PRINBAL(LOOP);
                CONTINUE = 0;
            END;
        END;
    END;

```

```

END;
RUN;
%MEND;
%TESTFILE

PROC MEANS DATA=ARRAYTSFLAG_F; (6)
    VAR MOB_F PRINBAL_1 PRINBAL_F;
    TITLE "FORECLOSURES, JUNE2006 - APRIL 2009";
RUN;

```

- (1) Specifies the beginning and ending dates of the time series.
- (2) Reads in an external file (optional). The only requirement is that the data set contains the by variable in sort order.
- (3) Defines array variables based on the enumerated variables constructed in the time series.
- (4) Loops through the time series for each unique ID.
- (5) Boolean construct that performs an action when the condition is true.
- (6) Descriptive output at the discrete level – the intended goal!

CONCLUSION

This paper assumes that a time series file is a useful construct in analyzing data by discrete category over time. The time series macro accomplishes this aim, and can be discerned through a careful reading of the macro **TESTFILE**.

APPENDIX

```

/*
TITLE:  TIME SERIES MACRO
AUTHOR: S. HANSON
DATE:   JULY 2009
SOURCE:
REV:
*/

LIBNAME S '\\calsas01\Share\Scott\sasdata';

OPTIONS MACROGEN NOSYMBOLGEN COMPRESS=BINARY NOCENTER NOLABEL;

%LET OBS = 1000; /* TEST */

%LET ORIGVARS =    LOANUM DTFUNDMON ZIP;
%LET KEEPVARS =    DLQ MOB PRINBAL CREDLIMIT;

DM LOG 'CLEAR';

%MACRO RENAME(G,VARLIST);
    %LET VARCNT = 1;
    %DO %UNTIL(&VARCNT EQ 0);
        %LET CVAR = %SCAN(%STR(&VARLIST),&VARCNT);
        %IF &CVAR EQ %STR() %THEN %LET VARCNT = 0;
        %ELSE %DO;
            %LET VARCNT = %EVAL(&VARCNT + 1);
            &CVAR=&CVAR.&G
        %END;
    %END;
%MEND;

%MACRO LABEL(FIRST, LAST);
    %LET VARCNT = 1;
    %DO %UNTIL(&VARCNT EQ 0);
        %LET CVAR = %SCAN(%STR(&KEEPVARS.),&VARCNT);
        %IF &CVAR EQ %STR() %THEN %LET VARCNT = 0;
    %END;

```

```

        %ELSE %DO;
            %LET VARCNT = %EVAL(&VARCNT + 1);
            %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
            %DO I = 1 %TO &INTRV.;
                %LET DD = %EVAL(%SYSFUNC(INTNX(MONTH,&FIRST., &I.)) - 1);
                %LET YY = %SYSFUNC(YEAR(&DD.));
                %LET MMM = %SYSFUNC(PUTN(%SYSFUNC(MONTH(&DD.)),Z2.));
                %LET DT = &YY.&MMM.;
                LABEL &CVAR.&I. = "&CVAR.&I. AS OF &DT.";
            %END;
        %END;
    %END;
%MEND;
%MACRO INFILES(FIRST, LAST, FPREFIX);
    %GLOBAL DT INTRV;

    %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
    %DO I = 1 %TO &INTRV.;
        %LET YY = %SYSFUNC(YEAR(&DD.));
        %LET MMM = %SYSFUNC(PUTN(%SYSFUNC(MONTH(&DD.)),Z2.));
        %LET DT = &YY.&MMM.;
        &FPREFIX.&DT.
    %END;
%MEND;

%MACRO INARRAYFILES(FIRST, LAST);
    %GLOBAL DT;

    %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
    %DO I = 1 %TO &INTRV.;
        %LET DD = %EVAL(%SYSFUNC(INTNX(MONTH,&FIRST., &I.)) - 1);
        %LET YY = %SYSFUNC(YEAR(&DD.));
        %LET MMM = %SYSFUNC(PUTN(%SYSFUNC(MONTH(&DD.)),Z2.));
        %LET DT = &YY.&MMM.;
        S.AF&DT. (OBS=&OBS. IN=IN&I.
            &WHERE.
            KEEP=&ORIGVARS. &KEEPVARS.
            RENAME=(%RENAME(&I.,&KEEPVARS.)))
    %END;
%MEND;

%MACRO SUBS(FIRST, LAST);
    %LET INTRV = %EVAL(%SYSFUNC(INTCK(MONTH,&FIRST.,&LAST.)) + 1);
    %DO I = 1 %TO &INTRV. - 1;
        IN&I. OR
    %END;
    IN&INTRV.
%MEND;
%MACRO ARRAYTS(FDAY, END);
    DATA ARRAYTS;
        MERGE %INARRAYFILES(&FDAY., &END.)
        ;
        BY LOANNUM;
        IF %SUBS(&FDAY., &END.)
        ;
        %LABEL(&FDAY., &END.)
        ;
    RUN;
%MEND;

/*
%ARRAYTS('01JUN2006'D, '01SEP2006'D)
*/

```

RECOMMENDED READING

Carpenter's Complete Guide to the SAS Macro Language, Second Edition, Art Carpenter, SAS Institute (2004).

CONTACT INFORMATION

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

Name: Scott Hanson
Enterprise: Bank of America
Address: 4500 Park Granada, Mail Code CA7-910-02-27
City, State ZIP: Calabasas, CA 91032
Work Phone: 1.818.225.3375
E-mail: scott.p.hanson@bankofamerica.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 trademarks of their respective companies.