Handling Non-Date Dates in SAS
Gary Seale and Anna Watson
Federal Housing Finance Board, Washington, DC

ABSTRACT

The SAS® system provides numerous methods for handling date and time values. However, we often work with datasets that represent date values as either character or numeric fields, and we need the ability to perform date and time calculations on these values just as we would for SAS date values. Additionally, in much of our work we use VBA® programs to call SAS stored process, where date and time values are passed as parameters from Excel® into macro variables that must subsequently be evaluated by SAS in order to subset the data. This paper demonstrates a few of the methods available in SAS for dealing quickly with these non-date variables. It shows how SAS can be used to take a single date macro value, convert it to a SAS date value, create new macro variables with the other required dates, and finally convert numeric date fields from a SAS dataset into SAS dates to perform calculations.

INTRODUCTION

Let’s suppose that we need to calculate the net interest margin for a group of financial companies. We want to be able to perform this calculation quarterly, and the only variable we would like to have to provide is the quarter end date for the desired quarter. Our dataset has information on the net interest income and average daily balance of earning assets for each month in the year. The net interest margin is defined as net interest income divided by average earning assets. However, for the quarterly net interest margin we need to divide the sum of monthly net interest income by the day-weighted average of the monthly average of earning assets. Using the first quarter of 2006 as an example, we need to divide the sum of net interest income for January, February and March by the following:

\[
\frac{\text{#Days}_{\text{Jan}} \times \text{AvgEA}_{\text{Jan}}} {\text{#Days}_{\text{Jan}} + \text{#Days}_{\text{Feb}} + \text{#Days}_{\text{Mar}}} + \frac{\text{#Days}_{\text{Feb}} \times \text{AvgEA}_{\text{Feb}}} {\text{#Days}_{\text{Jan}} + \text{#Days}_{\text{Feb}} + \text{#Days}_{\text{Mar}}} + \frac{\text{#Days}_{\text{Mar}} \times \text{AvgEA}_{\text{Mar}}} {\text{#Days}_{\text{Jan}} + \text{#Days}_{\text{Feb}} + \text{#Days}_{\text{Mar}}} 
\]

Thus, the daily average of earning assets for each month must be multiplied by the number of days in that month, summed for the quarter, and then divided by the total days in the quarter. The dataset Finance1 contains the data:

<table>
<thead>
<tr>
<th>Finance1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompanyID</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
The period field is a numeric field representing the date as the four-digit year followed by the two-digit month. What we need is a way to subset the records in the first quarter of 2006 based on the value of the macro input parameter, and then to convert the numeric date variable date into a SAS date so that we can calculate the number of days in each month.

CONVERTING THE INPUT PARAMETERS

The input parameters pass to the SAS program from Excel as macro variables, which are simply evaluated as text. They could also simply be coded macro variables. The following code is used to convert the quarter-end date variable into a SAS date value and then uses this code to determine the ending date of the previous quarter. These two variables can then be used to subset the data.

**CONVERTING INPUT PARAMETERS:**

```sas
%LET rptdatein=31Mar2006;
*Processbody;
%LET rptdate="&rptdatein"d;
DATA _null_;
  CALL SYMPUT('Tempdate',PUT(INTNX('Month',"&rptdatein"d,-3,'End'),date9.));
RUN;
%LET BegDate="&Tempdate"d;
%PUT _user_;
```

This code uses a %LET statement to enclose the date received from Excel in quotation marks and place the ‘d’ at the end. This is still a text variable, but after it is resolved by the macro processor it will be interpreted by the program compiler as a date value. By using the CALL SYMPUT function within a data step, we can use the INTNX function to create a third macro variable that is a string containing the last date of the previous quarter. The date9 format is used to ensure the macro variable resolves to 31DEC2005. Finally, the last %LET statement creates the variable BegDate by putting Tempdate into date format. The %Put statement at the end prints all user created macro variables to the SAS log.

SUB SETTING THE DATA AND PERFORMING CALCULATIONS

The final step is to create a SAS date field from the numeric date field, subset the data, and perform the required day count calculations. The following code uses the MDY and INTNX functions to create a new field called Date, and then filters the data using the macro variables created in the step above.
CREATING A DATE FIELD AND FILTERING DATA:

```sas
DATA Work.One;
SET Finance1;
Datein=MDY(INPUT(SUBSTR(PUT(Period,6.),5,2),2.),1,INPUT(SUBSTR(PUT(Period,6.),1,4),4.));
Date=INTNX(‘Month’,datein,0,’End’);
IF &BegDate<Date<=&RptDate;
RUN;
```

The middle lines of code in this program contain the MDY and INTNX functions. While we can’t be sure on what day any given month will end, we are absolutely sure on what day each month begins – the first day. Thus, a temporary variable named Datein is created and set equal to the first day of each month. The INTNX function is used with an increment of zero to create a variable called Date for the last day of each month. This variable is a SAS date. The final IF statement filters the data so that only records in the first quarter of 2006 are included.

There are a number of ways to finish the calculation. The code below presents one example of using PROC SUMMARY to finish the calculation.

```sas
PROC SUMMARY EXAMPLE:
DATA Work.Two;
SET Work.One;
Weight=date-datein+1;
WAvgEA=AvgEA*weight;
RUN;

PROC SUMMARY Data=Two SUM NOPRINT;
VAR weight NII WAvgEA;
CLASS CompanyID;
OUTPUT OUT=Sumout SUM=Sweight SNII SWAvgEA;
RUN;

DATA NIMargin
SET Work.Sumout;
IF _TYPE_ = 1;
NIM=SNII/(SWAvgEA/Sweight);
RUN;
```

CONCLUSION

Using the MDY function with the INTNX function provides a quick method for creating date fields from non-date fields. This method works equally well with character and numeric values. Macro variables can also be easily reformatted using a simple %LET.
statement or by using a data step with a CALL SYMPUT function. These techniques are especially useful in performing date calculations on SAS datasets that have no date fields.

CONTACT INFORMATION

Gary Seale
Federal Housing Finance Board
sealeg@fhfb.gov
202-408-2813

Anna Watson
Federal Housing Finance Board
watsona@fhfb.gov
202-408-2982

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.