Paper ###-2015

Let Dates Drive Your Data: A Simple Primer Setting up Macro Dates

Crystal Carel, MPH, Baylor Scott & White Health, Dallas, TX

ABSTRACT

Dates have been and continue to be an obstacle in data management and reporting among SAS programmers. Almost all data management processes and reporting rely on dates, whether it be admission, procedure, discharge, date of service, occurrence, billed, or paid date in health care situations and other business environments. Setting up dates to be automated macros provides a set standard against how data manipulation and reporting will be handled. The same methodology will be in place every time in the SAS session allowing for consistent results over time.

This paper will provide a SAS programmer basic instruction with the code needed to set up and create their own macro dates in order to help automate and drive their report. We will discuss and demonstrate the different options available to set dates and customize formats for data management and reporting. A demonstration of setting up macro dates will be provided with interpretation of the different elements a SAS programmer can change to customize their date macro. Using macro dates in code will help reduce time spent on manual coding changes and data errors due to forgetting to update code.

INTRODUCTION

You have a report that you need to run on a schedule (yearly, monthly, weekly, etc). However, having to manually change the date every time you run the report is not only time consuming but also menacing especially if you forget what time period should be included on the report. Dates that are data driven are necessary if the process will eventually be put into a scheduler or automated process. Dates in SAS (without formats) are stored in a numeric 5 digit format that doesn't make much – so formats are a must throughout the process in order for the date to be presented in a manner one understands. (Run the code in the next section that creates DATA A without the format to see original SAS date outcome).

SIDE NOTE: This paper discusses dates and not date/time stamps. However if you have a date/time stamp variable and are needing to have macros based on your date alone, you can convert the date/time stamp like this:

```
DATA TIMESTAMP_CONVERT;
FORMAT NEWDATEVAR DATE9.;
NEWDATEVAR = DATEPART(OLD_TIMESTAMP_DATE);
RUN;
```

STARTING WITH A DATE

To get started out, we will first create a data set with today's date:

```
DATA A;
FORMAT CALDATE DATE9.;
CALDATE = TODAY ();
RUN;
```

We now have today's date (14OCT2015) as a variable called **CALDATE**. Your data may be stored differently or you may want to report the dates in a different format than it was stored. Once you have a date, the options are endless in how you can format that date from bringing in data, throughout the data management process, and in reporting.

Next we are going to explore various types of ways that (14OCT2015) aka **CALDATE** can be represented. By using PROC SQL, we will create macros with the particular date. These macros are helpful in data management when you are looking for data that was before, during, or after a certain time period. We can also use the dates later on in reporting.

The PROC SQL code below will not produce any visible output however we will get that in the next step. Also note that the INTO statement is necessary to name your macros. I tend to name macros up date with the AS statement then call the date macro the same name in the INTO statement as good practice.

```
PROC SQL NOPRINT;
SELECT  INTNX("DAY",   CALDATE, 0,"E")  FORMAT=date9.       AS dateA
       ,INTNX("MONTH", CALDATE, 0,"B")  FORMAT =mmddyy10.   AS dateB
       ,TRIM(left(PUT(CALCULATED dateA,worddate20.)))       AS dateC
       ,(CALCULATED dateA)              FORMAT =worddate3.  AS dateD
       ,(CALCULATED dateA)              FORMAT =year4.      AS dateE
       ,INTNX("YEAR" , CALDATE,  0,"B") FORMAT =date9.      AS dateF
       ,INTNX("YEAR" , CALDATE,  7,"B") FORMAT =date9.      AS dateG
       ,INTNX("MONTH", CALDATE, -2,"B") FORMAT =yymmn6.     AS dateH
       ,INTNX("MONTH", CALDATE, -2,"E") FORMAT =monyy7.     AS dateI
       ,INTNX("MONTH", CALDATE,-13,"B") FORMAT =date9.      AS dateJ
INTO   :dateA,:dateB,:dateC,:dateD,:dateE,:dateF,:dateG,:dateH,:dateI,:dateJ
FROM A;
QUIT;
```

In order to see how **CALDATE** (14OCT2015) has changed, we need to visualize the data. We can do that by using the %PUT function to see how the date was formatted.

```
%put dateA = &dateA;
%put dateB = &dateB;
%put dateC = &dateC;
%put dateD = &dateD;
%put dateE = &dateE;
%put dateF = &dateF;
%put dateG = &dateG;
%put dateH = &dateH;
%put dateI = &dateI;
%put dateJ = &dateJ;
```

In order to see the output, open up the LOG window and scroll until you see your %PUT statements.

```
%put dateA = &dateA;
dateA = 14OCT2015

%put dateB = &dateB;
dateB = 10/01/2015

%put dateC = &dateC;
dateC = October 14, 2015

%put dateD = &dateD;
dateD = Oct

%put dateE = &dateE;
dateE = 2015

%put dateF = &dateF;
dateF = 01JAN2015

%put dateG = &dateG;
dateG = 01JAN2022

%put dateH = &dateH;
dateH = 201508

%put dateI = &dateI;
dateI = AUG2015

%put dateJ = &dateJ;
dateJ = 01SEP2014
```

BREAKDOWN OF THE CALCULATION

We are able to create and calculate any date imaginable just using today's date. To better explain what the calculations are doing let's break up some and explain step by step:

```
INTNX("DAY",    CALDATE, 0,"E")        FORMAT=date9.          AS dateA
```

1. INTNX is the function being used which increments dates/times. (See REFERENCES #1 for background.)
2. An interval time period – day, week, month, year, qtr – always has to be in "quotes" (single or double).
3. This is our variable that the calculation will use as a 'base'. We may want the quarter associated, year, month, or three months prior to this particular date. We can calculate all of these based off our 'base' date.
4. This number (presented as 0 above) can be positive (+), negative (-), or 0.
    - Positive means future time period. You do not have to use the positive (+) sign, but can. Ex: **dateG**.
    - Negative means prior time period. You must use the negative (-) sign. Example: **dateH**.
    - 0 means current time period. Example: **dateA**.
5. This defines the time period interval you are wanting: E=END, B=Beginning, S=Same Day, M=Middle, etc.
6. FORMAT = this is the format you want displayed. There is a whole host of options. (See REFERENCES #2.)
7. Name of the new variable we are creating from the calculation. This variable can be used later on in date macros or reporting to show the new date.

EXAMPLE EXPLANATIONS

```
INTNX("DAY",    CALDATE, 0,"E")        FORMAT=date9.          AS dateA
```

For our example above, we are wanting to get the day from **CALDATE** that is at the END in the DATE9. format and it'll be called **dateA**. Thus our output looks exactly the same as what we started with 14OCT2015.

```
INTNX("MONTH", CALDATE, 0,"B")        FORMAT =mmddyy10.      AS dateB
```

For another example (**dateB**), we want the month from **CALDATE** that is during the same month (indicated by 0), but at the BEGINNING instead using a different format of MMDDYY10. so we get 10/01/2015 – however we could get 01OCT2015 if we used the DATE9. format.

We can even pick off just a month name or a year instead of an entire date.

```
(CALCULATED dateA)                    FORMAT =year4.         AS dateD
```

We've already got a **dateA** that was 14OCT2015, however we may want to know the year only. To do this, we can use the CALCULATED function within parenthesis and add a format statement.

```
INTNX("YEAR" , CALDATE,  7,"B")       FORMAT =date9.         AS dateG
```

Also we can calculate previous dates or future dates. From our CALDATE, we can grab the year (2015) and then add 7 more (2022) and use the very beginning of the year that will be displayed as 01JAN2022 with the DATE9. format.

USING DATE MACRO IN A DATASET

Let's use the macro dates you've created in a dataset to see how we can use this in your programming. We create a dataset called A_PATIENTS – you can copy and paste this code to follow along with the output.

```
DATA A_PATIENTS;
   INFILE DATALINES4 DLM='#';
   FORMAT DISCHARGEDATE DATE9.;
   INPUT NAME & $16.
              ACCOUNT $
              CHARGES
              LOCATION $
              DISCHARGEDATE DATE9.;
DATALINES;
Apollo, Alfred# 1111# 332#   HHP# 30JAN15
Bush, Baley#    2222# 10732# HHP# 18JUN14
Cappi, Chris#   3333# 6433#  HHP# 20MAR15
Doug, Devon#    4444# 982#   BHV# 18SEP14
Eggle, Edgar#   5555# 632#   BHV# 10JAN15
Finnel, Finn#   6666# 2353#  BUC# 16FEB15
Guaz, George#   7777# 225#   BHV# 02FEB15
Harris, Hubert# 8888# 1114#  BUC# 15APR14
Ingels, Ingrid# 9999# 8734#  TEM# 19JUN14
Jones, Jacob#   1010# 436#   TEM# 18DEC14;
RUN;
```

Using our dataset A_PATIENTS – we want to see patients who had a DISCHARGEDATE this year which is 2015. In our dataset, we have our dates as DATE9. format. We can change our format in the dataset or we can use the DATE9. format and a date macro we already set up based off today's (14OCT2015) date.

```
INTNX("YEAR" , CALDATE,  0,"B")       FORMAT =date9.        AS dateF
```

Remember our **dateF** macro – it pulls the year based off today's date and looks at the very beginning of the time period (which is January 1) and outputs the data in a DATE9. format and our output was 01JAN2015.

We can use this macro to find out patients who had a DISCHARGEDATE this year.

```
DATA B_THIS_YEAR;
SET A_PATIENTS;
if DISCHARGEDATE GE "&dateF"d;
RUN;
```

We can use this macro to find out patients who had a DISCHARGEDATE in a prior year.

```
DATA C_PRIOR_YEARS;
SET A_PATIENTS;
if DISCHARGEDATE LE "&dateF"d;
RUN;
```

When using date macros with your code – you may need to use "quotes" and also the letter 'd' which is telling SAS it's a date.

PUSHING DATE MACROS FURTHER

Once you get the hang of creating a few date macros – you can do a lot more to get the 'pieces' of the date that you want. To understand what I mean, use the following code based off our original dataset A.

Again the PROC SQL code below will not produce any visible output however we will get that in the next step.

```
PROC SQL NOPRINT;
SELECT  INTNX("MONTH",CALDATE,  3,"E")  FORMAT=date9.       AS dateA
       ,STRIP(PUT(CALCULATED dateA,year4.))                 AS YEAR
       ,INTNX("MONTH",CALDATE,-14,"B")  FORMAT=worddate9.   AS C1
       ,COMPRESS(PUT(CALCULATED C1,worddate9.))             AS C2
       ,STRIP(PUT(CALCULATED C1,year4.))                    AS C3
       ,COMPBL(CAT(CALCULATED C2, CALCULATED C3))           AS C4
       ,INTNX("month",CALDATE,9,"B")    FORMAT=worddate9.   AS T1
       ,COMPRESS(PUT(CALCULATED T1,worddate9.))             AS T2
       ,STRIP(PUT(CALCULATED T1,year4.))                    AS T3
       ,COMPBL(CAT(CALCULATED T2, CALCULATED T3))           AS T4
INTO    :dateA,:YEAR,:C1,:C2,:C3,:C4,:T1,:T2,:T3,:T4
FROM A;
QUIT;
```

In order to see how **CALDATE** (14OCT2015) has changed, we need to visualize the data. We can do that by using the %PUT function to see how the date was formatted.

```
%PUT dateA =&dateA;
%PUT YEAR  =&YEAR;
%PUT C1 =&C1;
%PUT C2 =&C2;
%PUT C3 =&C3;
%PUT C4 =&C4;
%PUT T1 =&T1;
%PUT T2 =&T2;
%PUT T3 =&T3;
%PUT T4 =&T4;
```

We can also combined date macro variables in order for reporting titles:

```
%LET title1=&C4;
%LET title2=&T4;
%LET measureperiod=&title1-&title2;
%PUT measureperiod=&measureperiod;
```

In order to see the output, open up the LOG window and scroll until you see your %PUT statements.

```
%PUT dateA =&dateA;
dateA =31JAN2016

%PUT YEAR =&YEAR;
YEAR =2016

%PUT C1 =&C1;
C1 =   August

%PUT C2 =&C2;
C2 =August

%PUT C3 =&C3;
C3 =2014

%PUT C4 =&C4;
C4 =August 2014

%PUT T1 =&T1;
T1 =     July
```

```
%PUT T2 =&T2;
T2 =July

%PUT T3 =&T3;
T3 =2016

%PUT T4 =&T4;
T4 =July 2016

%PUT measureperiod=&measureperiod;
measureperiod=August 2014-July 2016
```

EXAMPLE EXPLANATIONS

All of the macro dates we just created were based off of CALDATE (14OCT2015) which is today's date. I'll walk through each one of these macros as we are expanding our understanding further to see some possibilities of date macros and how we can manipulate dates.

```
INTNX("MONTH",CALDATE,  3,"E")    FORMAT=date9.        AS dateA
STRIP(PUT(CALCULATED dateA,year4.))                    AS YEAR
```

We created dateA looking at the CALDATE month, adding 3 months, and reporting the ending time period of that month thus the output is 31JAN2016. Using dateA macro date as the base for YEAR, we are stripping off the month and day to only get the year (2016) set as the YEAR macro.

```
INTNX("MONTH",CALDATE,-14,"B")    FORMAT=worddate9.    AS C1
COMPRESS(PUT(CALCULATED C1,worddate9.))                AS C2
STRIP(PUT(CALCULATED C1,year4.))                       AS C3
COMPBL(CAT(CALCULATED C2, CALCULATED C3))              AS C4
```

The purpose for this section is to create a MONTH YEAR variable that can be presented for reporting as a title. However I need all leading and trailing blanks/spaces to be removed so the title looks perfect.

- For C1, we take the CALDATE month and look back -14 months to get the month name alone.
- C2 takes the month C1 calculated and compresses out the leading and trailing blanks/spaces. If you look very closely at the output of C1 and C2, you can leading blanks/spaces.
- C3 is doing the same thing as YEAR did above. C3 is stripping off the month and day to grab only the year.
- C4 is concatenated the month and year we calculated from C2 and C3 together, while also removing leading and trailing blanks/spaces. You'll see there is a space between the month and year as we made that space in our calculation. If we instead had: `COMPBL(CAT(CALCULATED C2,CALCULATED C3))AS C4` then the space between the month and year would not be there.

We do the same steps for T1-T4 but for a future time period.

For the combined date macro variables for reporting titles, we set each final date macro (the month with year) to its own macro then we create a new macro called MEAUSREPERIOD that displays: `August 2014-July 2016` You'll can add more space between the "-" (dash) when you set the MEASUREPERIOD macro.

CODE:

```
%LET TITLE1=&C4;
%LET TITLE2=&T4;
%LET MEASUREPERIOD=&TITLE1-&TITLE2;
%PUT MEASUREPERIOD=&MEASUREPERIOD;
```

LOG:

```
%PUT measureperiod=&measureperiod;
measureperiod=August 2014-July 2016
```

CONCLUSIONS

Dates are and will continue to be one area of programming that can break your code or even worse change the final data and outcomes. By setting date macros, your programs can be guarantee to be based off the same parameters each time the code is ran which decreases errors by manually manipulation. When creating date macros, be aware of leading and trailing blanks as this often causes much headache.

Be aware, when you set up your data, reporting, etc. based on data driven dates – the date parameters will change if it is October 31st vs. November 1st, same for different quarters or December 31st vs January 1st. Even though the dates are 1 day apart, it will change your date parameters. If you are developing a report – you may need to change up the macros to get development data and change back for final reporting. Regardless I suggest always keep comments of what the date parameter should be for each macro in order for documentation within the program or for various testing, so you always know what the original date parameter should be set.

REFERENCES/RECOMMENDED READING

1. http://support.sas.com/documentation/cdl/en/etsug/60372/HTML/default/viewer.htm#etsug_tsdata_sect038.htm

2. http://www.okstate.edu/sas/v8/saspdf/ets/chap3.pdf

3. http://www.amadeus.co.uk/sas-training/tips/1/1/15/getting-the-most-out-of-the-intnx-function.php

CONTACT INFORMATION

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

Crystal Carel
Baylor Scott & White Health
8080 N. Central Expressway, Suite 500
Dallas, TX 75206
214-265-3674
Crystal.Carel@baylorhealth.edu