Paper 117-2011

When "Anydate" Doesn't Mean Any Date: A Macro Solution

Charlotte King, Alberta Health Services, Edmonton, AB
John Fleming, Alberta Health Services, Edmonton, AB

ABSTRACT

In our work, we often receive administrative and research data in Microsoft Excel workbooks. This data is used for developing standardized approaches to administrative reporting and for research projects. A lot of it contains dates that don't conform to either Excel or SAS® date formats, and are rarely entered consistently in the Excel workbook. In the past, a secretary would spend hours going through these workbooks reformatting dates so they were in a form that SAS could understand.

In SAS®9, SAS introduced the ANYDATE INFORMAT that can read data in a variety of date formats. This has its limitations, too. To accommodate these limitations, we developed a SAS macro. With our SAS macro, we have converted reading dirty date data from non-SAS sources from a labor-intensive process into an easy routine task, saving hours of time and reducing data entry errors resulting from the manual re-entry of dates. This macro is robust and expandable to a wide variety of date formats that SAS programmers might encounter.

INTRODUCTION

Receiving data in Excel workbooks is a daily reality for many SAS® programmers. The data we receive is used for developing standardized approaches to administrative reporting and in research projects, and therefore contains multiple columns of dates. Despite our best efforts to encourage data providers to use only one date format, the Excel workbooks we receive contain date data that is rarely entered consistently within any one column. Also the dates frequently do not conform to any Excel or SAS date format.

Prior to SAS®9, someone would spend hours going through the workbook reformatting dates so they were consistent within the columns. While it is possible to write an Excel VBA macro to do this, our team members were exclusive SAS users and did not have the skills needed.

At a SAS global forum, a team member learned about the ANYDATE INFORMAT and suggested it as a solution to the issue of reading in inconsistent dates. Given the extremely dirty nature of the dates in our data, we found that ANYDATE has its limitations too.

In order to solve the issues with our date data, we went through a number of steps, which we present here, in the development of a macro to work around shortcomings in the ANYDATE INFORMAT. This macro is expandable and can accommodate date issues faced by us in the future and by other programmers.

STEP 1

The ANYDATE INFORMAT is used to read the date data from the Excel workbook. This informat was introduced with SAS9 and reads in data using a number of common SAS date informats (DATE, DATETIME, YYMMDD etc…). If a date in the data corresponds to one of these informats, SAS will read the value in as a SAS date value. Otherwise, SAS sets the date value to missing in the resulting SAS data set. To accommodate ambiguous dates such as 01-02-03--that can be read in several different ways including January 2, 1903, and February 1, 2003—the DATESTYLE option is used to specify the order of month, day, and year.

```
options datestyle=locale;
   data dates1;
      set dates;
      datevar1=input(Fn[1],anydtdte.);
      format datevar1 date9.;
   run;
```

To determine the number of dates not read correctly, we use the FREQUENCY PROCEDURE to determine the proportion of dates that are missing. We then use the PRINT PROCEDURE to view the original date input. This is our standard check after each step.

[1] Fn is a default name for Excel columns where there is no header row, with n being an integer greater than or equal to 1.

```
proc freq data=dates1;
   tables datevar1 / missing;
run;

proc print data=dates1 (where=(missing(datevar1)));
   var Fn;
run;
```

Fn
mar 9,1949
apr-18-1937
oct 16-1938
apr,22,1935
may12-71
dec -6-1977
sep 1 1953
nov/6/1936

Table 1. Sample of date formats that the ANYDATE INFORMAT returns as missing.

We found that the ANYDATE INFORMAT does not have the ability to deal with dates with inconsistent separators.

STEP 2

We remove unwanted spaces and punctuation marks from the dates in our source data and try the ANYDATE INFORMAT again.

```
options datestyle=locale;
   data dates2;
      set dates;
      Fn=compress(compress(lowcase(Fn), ,'s'), ,'p');
      datevar1=input(Fn,anydtdte.);
      format datevar1 date9.;
   run;
```

We perform our check and found that there are still many missing date values.

Fn
mar91949
apr181937
oct211932
dec271969
mar81933
jan241933
oct151938
jan251953

Table 2. Sample of dates that the “anydate” informat returns as missing after eliminating unwanted spacers.

At this point, we discovered that SAS formats do not always have a corresponding informat for reading data, for example, there is no WORDDATE INFORMAT, despite there being a WORDDATE FORMAT.

STEP 3

We turned to the SAS online documentation and SAS Global Forum proceedings to learn how to create informats from existing SAS formats. The key is to create a SAS data set that contains the following columns: 1) the SAS numeric date values, 2) the corresponding formatted date values, 3) the format name and 4) the type of format. The format name must comply with SAS format naming rules, that is, they must be less than eight characters and cannot end with a number. The type of format is set to ‘I’ for informats. A format that can handle other values not specifically mentioned in the format definition has a line that looks like "other = ' '". Failure to include this line causes SAS to generate an error message whenever it encounters a value not included in the informat definition. The last four lines of the DATA step create this “other” line in the format file, and the informat will write a blank each time it encounters a date that is not otherwise defined in the informat definition.

Once the data set is complete, it is simply a matter of using the FORMAT PROCEDURE to create the new informat.

```
data infmt1;
   retain fmtname "worddtin" type "I";
   do label="1jan1900"d to "31dec2025"d;
      start=put(label,worddate.);
      start=trim(left(start));
      start=compress(compress(lowcase(start), ,'s'), ,'p');
      output;
   end;
   hlo='O';
   label=.;
   start=' ';
   output;
run;

proc format cntlin = infmt1;
run;
```

Next, we read in our data using this new informat and the ANYDATE INFORMAT.

```
options datestyle=locale;
   data dates3;
      set dates;
      Fn=compress(compress(lowcase(Fn), ,'s'), ,'p');
      datevar1=input(Fn,anydtdte.);
      if missing(datevar1) then datevar1=input(Fn,worddtin.);
      format datevar1 date9.;
   run;
```

We checked again and found that there are still some dates in our source data that could not be read into SAS because SAS does not have a corresponding date format that can be changed into an informat.

STEP 4

We returned to the SAS Global Forum proceedings to learn how to create custom date formats that could be turned into informats.

Within PROC FORMAT, you can create any custom date format using the PICTURE statement. Simply assign a format name, specify the length of the format, design the look of the format by using the date directives found in the online documentation and specify the DATATYPE. For this example, we create a format for dates in a month-day-year order with no leading zero on the day. Once the format has been created then the informat is created as before.

```
proc format;
   picture mondyyyy (default=11) other ='%b %d %Y' (datatype=date);
run;

data infmt3;
   retain fmtname "mondyyyy" type "I";
   do label = "1jan1900"d to "31dec2025"d;
      start = put (label,mondyyyy.);
      start = trim ( left (start) );
      start=compress(compress(lowcase(start), ,'s'), ,'p');
      output;
   end;
   hlo='O';
   label=.;
   start=' ';
   output;
run;

proc format cntlin=infmt3;
run;

options datestyle=locale;

   data dates4;
      set dates;
```

```
   f2=lowcase(compress(compress(f2, , 'p'), , 's'));
   f5=lowcase(compress(compress(f5, , 'p'), , 's'));

   ref_date=input(f2, anydtdte.);
   dob=input(f5, anydtdte.);

   if missing(ref_date) then ref_date=input(f2, worddtin.);
   if missing(dob)      then dob=input(f5, worddtin.);
   if missing(ref_date) then ref_date=input(f2, mondyyyy.);
   if missing(dob)      then dob=input(f5, mondyyyy.);

   format dob ref_date date9.;

run;
```

We then did our check for missing values as before, and found that there were still some dates that are problematic, namely, dates with two digit years and days with leading zeroes.

Fn
jan041976
jun1533
sep1167
jul041943
dec0244
3301934
oct0462
jan1159

Table 3: Sample of data with two digit years and days with leading zeroes.

STEP 5

We create a second picture format using a lower case ‘y’ for two digit years and adding a zero before y to indicate leading zeroes.

```
proc format;
picture mondyy (default=9) other ='%b %d %0y'    (datatype=date);
run;
```

Because we are using two digit years, this limited us to a one hundred year interval. For our purposes, we use the period January 1, 1920, to December 31, 2019, however this can be set to cover other time intervals as needed.

```
data infmt4;
   retain fmtname "mondyy" type "I";
   do label = "1jan1920"d to "31dec2019"d;
      start = put (label,mondyy.);
      start=compress(compress(lowcase(start), ,'s'), ,'p'); output;
   end;
   hlo='O';
   label=.;
   start=' ';
   output;
run;

proc format cntlin=infmt4;
run;

data dates5;
   set dates;

   f2=lowcase(compress(compress(f2, , 'p'), , 's'));
   f5=lowcase(compress(compress(f5, , 'p'), , 's'));

   ref_date=input(f2, anydtdte.);
   dob=input(f5, anydtdte.);
```

```
   if missing(ref_date) then ref_date=input(f2, worddtin.);
   if missing(dob)      then dob=input(f5, worddtin.);
   if missing(ref_date) then ref_date=input(f2, mondyyyy.);
   if missing(dob)      then dob=input(f5, mondyyyy.);
   if missing(ref_date) then ref_date=input(f2, mondyy.);
   if missing(dob)      then dob=input(f5, mondyy.);

   format dob ref_date date9.;

run;
```

We read in our data again and checked for missing values that indicate more date value issues. At this point, we decided a macro solution would provide more flexibility for addressing the many diverse and problematic date formats that data providers use.

STEP 6

There are two obvious advantages to our macro solution. First, we can quickly tailor the code to different variables in different data sources. For example, two variables like date of birth and referral date use essentially the same code; only the variable names change. A macro solution makes it easy to make these changes on the fly and improves maintainability. Second, a macro solution also allows for easier control of parameters like the start and end years for the 100 year time period when dates have a two digit year along with the most likely year used for dates that lack a year entirely. We can also easily switch the DATESTYLE option used with the ANYDATE INFORMAT. By using a macro solution, new date informats can also be quickly added as needed.

The DATA _NULL_ step in the macro is used to create new macro variables from the parameters defined in the %MACRO statement without keeping any of the intermediate values in memory.

A SAS CATELOG of the formats and informats is created in a separate program that is added to the macro with a %INCLUDE statement. The reasons for this are twofold: 1) as the number of formats and informats increased, the macro became unwieldy in terms of length. We foresaw a future when the length would start to seriously impact maintainability; 2) creating a SAS CATELOG allowed us to access the newly created formats for other projects and reports where the macro solution is not needed.

The order of IF, THEN statements can affect the results. For our purposes, the order given was appropriate.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                                                                     */
/*  MACRO NAME: DATES                                                  */
/*                                                                     */
/*  DATE CREATED: FEBRUARY 1, 2006                                     */
/*                                                                     */
/*  LAST UPDATED: FEBRUARY 23, 2011                                    */
/*                                                                     */
/*  DESCRIPTION:                                                       */
/*                                                                     */
/*    THIS MACRO READS DATES FROM NON-SAS SOURCE DATA, AND             */
/*    CONVERTS "MESSY" DATES ENTERED INTO THE SOURCE WITH NON-         */
/*    STANDARD DATE FORMATS INTO "CLEAN" DATES THAT SAS CAN            */
/*    USE AND WHICH ARE FORMATTED WITH STANDARD SAS FORMATS.           */
/*                                                                     */
/*  PARAMETERS:                                                        */
/*                                                                     */
/*    DSIN:          SOURCE DATA SET CONTAINING THE MESSY DATE        */
/*                   DATA.                                             */
/*                                                                     */
/*    DSOUT:         DESTINATION DATA SET FOR THE CLEANED UP          */
/*                   DATE DATA.                                        */
/*                                                                     */
/*    MESSY:         NAME OF SOURCE VARIABLE IN THE SOURCE DATA       */
/*                   SET WITH MESSY DATE DATA .                        */
/*                                                                     */
/*    CLEAN:         NAME OF VARIABLE IN DESTINATION DATA SET         */
/*                   WITH THE CLEANED UP DATE DATA.                    */
/*                                                                     */
/*    YEAR_START:    START YEAR FOR USE WITH DATES THAT HAVE A        */
```

```
/*                  TWO DIGIT YEAR.  (MACRO WILL CREATE A      */
/*                  CORRESPONDING YEAR_END MACRO VARIABLE BY   */
/*                  ADDING 99 TO THIS VALUE.)                  */
/*                                                             */
/*    MISSING_YEAR_START_DATE: START DATE FOR THE HUNDRED-YEAR */
/*                  INTERVAL FOR CUSTOM DATE FORMATS.  (MACRO  */
/*                  WILL CREATE A CORRESPONDING PARAMETER,     */
/*                  MISSING_YEAR_END_DATE, BY ADDING 99 YEARS  */
/*                  TO THIS DATE.                              */
/*                                                             */
/*    MISSING_YEAR: MOST LIKELY YEAR USED FOR DATES WHERE ONLY */
/*                  A DAY AND A MONTH ARE PROVIDED.            */
/*                                                             */
/*    DSTYLE:       DATE STYLE OPTION USED TO DETERMINE HOW TO */
/*                  READ AMBIGUOUS DAY, MONTH AND YEAR         */
/*                  COMBINATIONS.  ALLOWED VALUES ARE: MDY,    */
/*                  MYD, YMD, YDM, DMY, DYM AND LOCALE         */
/*                                                             */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* ----------------------------------------------------------- */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


%macro dates(inds, outds, messy, clean, year_start, missing_year_start_date,
missing_year, dstyle);

     data _null_;

       year_end = &year_start + 99;
       call symput('year_end',year_end);

       missing_year_end = mdy(month("&missing_year_start_date"d),
                              day("&missing_year_start_date"d),
                             year("&missing_year_start_date"d + 99));
       call symput('missing_year_end_date',missing_year_end);

     run;

     libname dateform "T:\date macro ";

     options yearcutoff=&year_start fmtsearch=(dateform) datestyle=&dstyle;

     %include "T:\date macro\date formats.sas";

     data &outds;
          set &inds;

          &messy=compress(compress(lowcase(&messy), , 's'), , 'p');

          &clean=input(&messy, worddate.);

          if missing(&clean) then &clean=input(&messy, anydtdte.);
          if missing(&clean) then &clean=input(&messy, worddatx.);
          if missing(&clean) then &clean=input(upcase(&messy), mondyyyy.);
          if missing(&clean) then &clean=input(&messy, mondyy.);
          if missing(&clean) then &clean=input(&messy, monddyyyy.);
          if missing(&clean) then &clean=input(&messy, monddyy.);
          if missing(&clean) then &clean=input(&messy, ddmon.);
          if missing(&clean) then &clean=input(&messy, dmon.);
          if missing(&clean) then &clean=input(&messy, montddyy.);
          if missing(&clean) then &clean=input(&messy, montdyy.);

          format &clean date9.;

     run;
```

```
%mend dates;

%dates(dates, dates1, F2, ref_dte, 1950, 1jan2006, 2006, locale);
%dates(dates1, dates2, F5, dob, 1900, 1jan2006,2006, locale);
```

CONCLUSION

With our SAS macro, we have converted reading dirty date data from a non SAS source from a labor intensive process into a routine task. This saves countless hours of effort, speeds the production of meaningful results, and reduces errors resulting from the manual re-entry of dates.

While we limited our macro to western date formats found in our data, this macro is robust and expandable to a wide variety of date formats, including non-western date formats, such as Japanese Nengo dates.

The limited number of formats with a corresponding date informat was surprising. Most SAS programmers, when first encountering dates, would naturally assume that for every date format there is a corresponding informat. We recommend, in a future release, that SAS address this limitation. Another limitation that SAS should address is the limited number of characters that can be used in a format name.

One weakness of our macro is that it does not error trap misspelled dates. For example, “January” misspelled as “Jamuary”. We plan to address this in the future. We are also considering adding a warning to quantify the number of ambiguous dates. This is mostly to help encourage data providers to be consistent and to stop using two digit years and numeric only dates. Given that order of the IF, THEN statements impacts the results, future versions of the macro will have the ability to select the order that is most correct for the data provided.

REFERENCES

Jonas V. Bilenas. Using SAS® Dates and Times – A Tutorial, SAS Global Forum 2007, Orlando, FL, Paper 226-2007.

Arthur L. Carpenter. Building and Using User Defined Formats, SAS Users Group International Conference 2004 Montreal, QC, Canada, Paper 236-29.

Arthur L. Carpenter. Looking for a Date? A Tutorial on Using SAS® Dates and Times, SAS Users Group International Conference 2005, Philadelphia, PA, Paper 255-30.

Steven First, Katie Ronk. SAS® Macro Variables and Simple Macro Programs, SAS Users Group International Conference 2005, Philadelphia, PA, Paper 130-30.

Wendi L. Wright. Creating a Format from Raw Data or a SAS® Dataset, SAS Global Forum 2007, Orlando, FL, Paper 068-2007.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Marcy Winget for her encouragement and support in completing this project, and Angela Bella for her invaluable help preparing the manuscript.

CONTACT INFORMATION

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

Name: Charlotte King
Enterprise: Alberta Health Services
Address: 15th Floor, Sunlife Place, 10123-99 Street
City, State ZIP: Edmonton, AB T6H 5S9
Work Phone: 780-643-4354
Fax: 780-643-4486
E-mail: charlotte.king@albertahealthservices.ca