

SAS® Doesn't Store Dates

Nat Wooding, J. Sargeant Reynolds Community College

ABSTRACT

Base SAS stores two types of variables: character strings and floating-point numbers. While a user can store a date value in a SAS data set using one of several methods, SAS does not recognize the value as being different from any other numeric or character variable. This paper explores the ways a date value can be stored and the ramifications of each storage method. In addition, it looks at some issues that are encountered when date values are read from data bases. This paper is intended for new SAS users and those who are not familiar with date handling in SAS.

INTRODUCTION

At the risk of committing hyperbole, I can't recall how many times I have seen a SAS/L post that includes a statement such as "I have a date stored in SAS in the format". My immediate reaction is "just what do you mean by that" since the actual storage method has major ramifications on how the post should be answered. In other cases, people are trying to read a file containing dates in various formats or they have received data from Excel® or a data base and are startled to find that SAS date functions do not seem to work. We will look at date storage methods and then spend a little time on working with dates from outside sources.

Sources of Confusion

One source of confusion in dealing with dates is the various ways that we use the word "format". In the same sentence, it could refer to the structure of file as well as the way the value of a variable appears in that file and also the way that SAS might display the value once the data have been read into SAS.

The Very Very Basics

SAS stores two types of variables: character strings of varying lengths and (forgetting the recently introduced Proc DB2) floating point numbers with a maximum length of 8 bytes. As an example, consider a data set with a variable of each type:

```
Cvar = '11/21/2016';
Nvar = 11212016;
```

Figure 1 shows the attributes of our data set.

Column Name	Type	Length	Format	Informat	Label
Cvar	Text	10			
Nvar	Number	8			

Figure 1 Attributes of variables in a data set

Note that although as we defined Nvar, it specified a value that we Westerners can recognize as a date but the data set attributes record nothing to distinguish it from any other number whether it be, say, my age, or the number of M&Ms in a bowl at a SAS Institute training class. I defy anyone to show me SAS printout of Proc Contents that shows a variable type that reads "Date".

While SAS doesn't recognize these variables as being dates, we who have been trained from infancy in the Gregorian system can, using our Mark 1 human brains, have been trained in the Gregorian system since childhood can automatically parse the structure of each data type and automatically recognize the parts and the whole as being a date composed of a month, a day, and a year, all calculated from a starting point some 2000 years ago.

So, we can store a date in either of these two ways: what is the difference and are we setting ourselves up for problems later on?

Some issues that can cause problems are

- 1) will we need to sort by date,
- 2) will we need to extract components of the dates for use in programs,
- 3) we may want to display the date in a different form such as a date with the month written out as a word, and
- 4) we might want to calculate the time difference between dates or find a date at some interval from our date.

If we can cross our hearts and swear that any of these events could never arise, then it doesn't matter how we store a date value. But, should any of the four be possibly lurking in the future, then we will have problems.

To illustrate, here are four scenarios that show the problems that are caused by storing dates in the ways shown so far.

Case 1: Sort by date

Our code:

```
Data Case_1a;  
    input Cvar $10.;  
cards;  
11/21/2016  
01/01/1999  
12/31/1999  
06/31/2017  
;  
Proc Sort;  
    by Cvar;  
run;
```

The result is shown in Figure 2.

	Cvar
1	01/01/1999
2	06/31/2017
3	11/21/2016
4	12/31/1999

Figure 2 Date values stored as a character variable

First, note that since this is a character variable, the sort reads the string from left to right so the dates are in month order but not by year. And, notice Obs 2. On my planet and in the Gregorian date system, there is no 31st day in June but SAS, not knowing this rule when dealing with character strings, could care less.

Similar problems can arise if we should write these dates as simple numbers and in that case, the number would have, repeat, have to be written in year-month-day order for sorts to work properly.

Case2: Extract the parts of the date

When dealing with characters, the Scan function used three times will readily break out the components but remember that we still have character strings. For example

```
Year = Scan ( Cvar , 3 , '/' );
```

Parsing a numeric version is a little more complex as in:

```
Nvar = 11082017;
Month = Int( Nvar / 1000000 );
Day = Int ( ( Nvar - Month * 1000000 ) / 10000 );
Year = Nvar - Month * 1000000 - Day * 10000 ;
```

This gives the desired results. Trust me.

Case3: Change appearance of a date such as changing the order of the year, month, and day.

As long as we are simply rearranging our parts, this is relatively simple if we have broken out the Year, Month, and Day. For Cvar to appear as Year/Month/Day, we simply concatenate the components

```
Month = Scan ( Cvar , 1 );
Day = Scan ( Cvar , 2 );
Year = Scan ( Cvar , 3 );
separator = '/';
NewCvar = Catx ( separator , Year , Month , Day ) ;
drop Separator;
```

Obviously, the parsing code will depend on the particular layout of the date variable and some forms are harder to handle than others. If our date appears as 03Nov2017 and we wanted it to appear as 2017/11/03, then would have to use a format or a series of If statements to convert the text month to a string of Arabic numerals.

For the Nvar, we simply add the parts:

```
NewNvar = Year * 1000000 + Month * 10000 + Day;
```

Case4: Calculate intervals or new dates related to our current one.

This is where things get really, really complicated. Assuming that our date is stored as character or number as in the examples of Case 2, we would have to go through the steps shown there to get the month, day, and year into their own numeric variables. Then we would have to count the intervening number of years, months, and days in order to figure out the result and this takes lots and lots of code with lots and lots of chances to make mistakes.

SAS to the Rescue

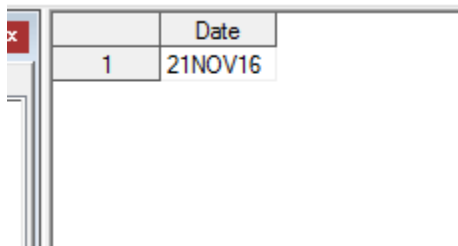
In the late 1970's, SAS introduced a method that would convert a date into the number of days before or after the fixed date or epoch, Jan 1, 1960. I'm not sure whether this feature was available in other languages back then but modern data bases use similar epoch-based methods of storing dates.

In order to store a date in this way in SAS, we have options shown in the following sections.

Make a direct assignment using a date literal value

```
Date = '21Nov2016'd;
```

the structure of this statement and the inclusion of the "d" tell SAS to evaluate the contents and assign the value 20779 to Date (when displayed in Best12. format). Were we to display it using the Date. format, we would see the value shown in figure 3.



	Date
1	21NOV16

Figure 3 Date stored as the number of days since 1/1/60 but using the Date. format for the display

Read it from a text file using the appropriate date informat.

```
Data InputStmt;  
input Date mmddyy10.;  
Format Date mmddyy10.;  
* Note that there are both a format and an informat named MMDDYY.  
Date2 = Date;  
* Date will be displayed using the mmddyy10. format and Date2 using  
Best12.;  
cards;  
11/21/2016  
11/21/16  
;
```

Note that I wrote the date two different ways and assigned a format to the input date variable but not to its copy, Date2.

When we examine this data set (Figure 4), we see that Date was read correctly in both cases and that the internally stored value is 20779.

	Date	Date2
1	11/21/2016	20779
2	11/21/2016	20779

Figure 4 Identical “SAS dates” displayed with different formats

To repeat myself, I have displayed two variables containing our original date value and assigned two different formats to them but the internal value has not been changed. Furthermore, any desired valid date format could be applied to either should the need arise and either variable could be operated on with any of the SAS Date functions. The SAS support website lists date formats but if I am searching for something special, I prefer the old Tech Support document TS 486, alas, abandoned by the Institute but still available on SASCommunity.ORG. It is especially good since the formats are grouped by topic such as date and not a simple alphabetic order. It is very easy to glance over when searching for a particular format. Figure 5 shows a sample of date and time functions from TS-486.

```

TIMEw.d      51745 | time8. | 14:22:25
              51745.23 | time11.2 | 14:22:25.23
TODw.        time portion from a datetime value
              956978640 | tod8. | 3:24:00
VMSTIMEF.*   [VMS] datetime value in VMS date and time format
WEEKDATEw.   day name, month name, day, and year from a date value
              10848 | weekdate3. | Wed
              10848 | weekdate9. | Wednesday
              10848 | weekdate17. | Wed, Sep 13, 1989
              10848 | weekdate29. | Wednesday, September 13, 1989
WEEKDATXw.   day name, day, month name, and year from a date value
              10848 | weekdatx17. | Wed, 13 Sep 1989
              10848 | weekdatx29. | Wednesday, 13 September 1989
WEEKDAYw.    day of week [1-7] from a date value
              10848 | weekday1. | 4
WORDDATEw.   month name, day, and year from a date value
              10848 | worddate3. | Sep
              10848 | worddate9. | September
              10848 | worddate12. | Sep 13, 1989
              10848 | worddate18. | September 13, 1989
WORDDATXw.   day, month name, and year from a date value
              10848 | worddatx3. | Sep
              10848 | worddatx9. | September
              10848 | worddatx12. | 13 Sep 1989
              10848 | worddatx18. | 13 September 1989
YEARw.       year from a date value
              10848 | year2. | 89
              10848 | year4. | 1989

```

Figure 5 Sample formats from TS-486

This document depends on the user community continuing to update it and at this point, I have not checked to see if the most recently announced formats are included. On the plus side, when the users took over this document, they added a couple formats that were available but not documented.

We have discussed how by using a special assignment statement or an appropriate Informat on an Input statement, SAS can read a date value and store it internally creating a value that is an offset of Jan 1, 1960. Furthermore, this stored value can be displayed using an array of formats. That is just part of the usefulness of this storage method. The second is that we can manipulate dates and perform operations on them.

At the simplest, if we have two different dates, Date1 and Date2, and we need to know the number of intervening days, we simply write

```
Days = Date2 - Date1;
```

There are two further operations that we might want to perform, 1) extracting a component of the date such as the month, 2) finding a new date that is related to the current one by some time interval, and three, convert the date to the equivalent in some other calendar system such as the Julian date. To do these operations, we invoke the appropriate SAS function.

Figure 6 is a snip of some of the date functions, again from TS 486.

INTNX	(int,fr,num) adds 'num' 'int' intervals to starting 'fr' fr and to are sas dates, times, or datetimes; interval values are listed in SASLang, p560 and SASP222+, p59 (int,fr,num,align) 'align' parm added in Rel 6.11 and 6.09E; advances result to the 'beginning', 'middle', or 'end' of the interval
JULDATE	(d) julian date equivalent of the sas date d
JULDATE7&	(d) julian date (with a 4-digit year) equivalent of the sas date d
MDY	(m,d,y) sas date month, day, and year
MINUTE	(tdt) minute [0-59] from either sas time or sas datetime value tdt
MONTH	(d) month [1-12] from the sas date value d
QTR	(d) quarter of the year [1-4] during which sas date value d falls

Figure 6 Date and Datetime functions

The QTR function extracts the quarter while INTNX will add some number of intervals to a date. INTNX is extremely useful when you need to increment a date using some interval such as month. At this point, someone is bound to be squirming and saying “but what about time values”. Yes, SAS will store a date/time value as the number of seconds in relationship to 1/1/1960. Should you look closely at the functions and formats, you will see some that operate on date/time data; I’ve been lazy and just talked about dates but almost all that has been said could apply to date/time values. The one exception would be the early case of storing a date as a raw number resembling a date; this would be difficult to do if time were included.

For those wishing to see TS 486, the link is

http://www.sascommunity.org/wiki/TS_486_Functions,_Informats,_and_Formats#Dates_and_Times_formats

Otherwise, you may find functions documented in the Base SAS documentation.

Intermission

Up to now, we have looked at storing date values as character strings, numbers using some combination of month, day, and year, and, finally, as the number of days before or after January 1, 1960. We have examined the potential disadvantages of the first two storage methods and the advantages of the last. At this point, we will look briefly at reading dates into SAS from some other software system and external files.

Help: My Date is Not Computing

Suppose we read a date from a data base such as Oracle®. Oracle and many or all other data bases store dates as date time values based on whatever starting point the developers chose. Likewise, Excel uses such a system based on 1900.

In the Oracle-based student information system used at my college, many date variables such as birthdate that are brought into SAS appear in a date-time layout which SAS can display as

```
06Nov2017:00:00:00
```

Here, the variable is clearly seen to be a date time value but the time part is meaningless. When dealing with dates like these, it is often a good idea early in your program to have a statement like

```
birthdate = datepart ( birthdate);
```

and we will now have a value stored in days and not seconds.

Whenever you receive a new data set from some outside source, it would often be a good idea to check

the attributes of the data. If the example above was read from Oracle and stored in SAS and I am using Display Manager in Windows, I can use the View Columns feature on the SAS file explorer window to check them as shown in Figure 7. Of course, Proc Contents would give the same information or we can try printing a few observations using a one of the Date or Datetime formats and see whether the displayed values are meaningful.

Column Name	Type	Length	Format	Informat	Label
123. 67. birthdate	Num...	8	DATETIME20.	DATETIME20.	

Figure 7 Attributes of a variable stored in seconds

Based on the fact that the Format and Informat are for date/time variables, it is readily apparent that we have a numeric variable stored in seconds and not days. If we are working on purely calendar days and hours, minutes, and seconds are irrelevant, I would use the Datepart function to convert the variable to days. I would also issue a Format statement so that the values would display (remember, this has nothing to do with the way that they are stored) in a meaningful way.

Suppose I Am Reading a Column with More Than One Informat

This can occur when the data have been entered by various people who have their personal favorite method of writing dates. Fortunately, SAS offers both a date informat and a date/time informat that are very flexible in converting a variety of layouts. The description of a very useful date informat as described in TS-486 is shown in Figure 8.

```
9ANYDTEw.    date values of any of the following forms: DATE, DATETIME, DDMMYY,
              JULIAN, MMDDYY, MONYY, TIME, YYMMDD, or YYQ
```

Figure 8 Date layouts readable with ANYDTEw. informat

CONCLUSION

While it is common to use the phrase “SAS Date” and I confess to this expediency, there is no such thing as a SAS variable type. Date values most assuredly can be stored as character values or numeric strings written as a combination of month, day, and year, but I strongly suggest that this be avoided. Using appropriate input or assignment methods, SAS can read and store a date but internally it is stored like any other number. A number of formats and functions can be applied to the value to display it in a variety of ways or to manipulate it. Furthermore, those who import date values from other systems should be aware that they are often stored as the number of seconds from some fixed point and functions that are based on the number of days from January 1, 1960, will give wildly ridiculous results.

REFERENCES

SAS 9.2 Language Reference: Dictionary, Fourth Edition, SAS Institute, Cary, N.C. 2011

SASCommunity.Org, TS 486 Functions, Informats, and Formats (A user community maintained and updated copy of a technical support document that SAS Institute no longer offers)

ACKNOWLEDGEMENTS

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 registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author at:

Nat Wooding
Office of Institutional Effectiveness
J. Sargeant Reynolds Community College
1651 East Parham Road
Richmond, VA 23228
804-523-5884

Email: Nwooding@reynolds.edu
Or Nathani@verizon.net