Paper 255-30

Looking for a Date? A Tutorial on Using SAS® Dates and Times

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT

What are SAS date and time values? How are they used and why do we care? What are some of the more important of the many functions, formats, and tools that have been developed that work with these crucial elements of the SAS System?

This tutorial will answer these questions and many more. Starting with date and time constants and their representation in SAS and then expanding to functions and formats, the basics of the use of dates and times will be integrated with their use as part of both data tables and data displays. The discussion will include functions and formats that are new to Version 8 and SAS 9.

Also covered are picture formats and date directives, date scaling in SAS/GRAPH, shift operators in the INTNX and INTCK functions, and the use of the %SYSFUNC macro function.

KEYWORDS

date, time, format, INTNX, INTCK, date literal, shift operator, alignment options

INTRODUCTION

WHAT ARE SAS DATE AND TIME VALUES?
There are three primary ways of measuring time in the SAS System. These are known as DATE, TIME, and DATETIME values. Each is a numeric value that measures from an arbitrary starting point by counting the number of elapsed time units. While somewhat inconvenient for us as SAS programmers to get used to, this is great for the computer because all interval calculations become simple addition and subtraction operations.

DATE values are stored as the number of days that have elapsed since the start of time (January 1, 1960). TIME values are the number of seconds that have elapsed since midnight of the current day. When you need a finer scale than that offered by DATE values, the DATETIME value counts the number of seconds that have elapsed since midnight of January 1, 1960.

On July 28, 2004 at 11:32 a.m. the SAS DATE value was 16,280 days since January 1, 1960. It was also 41,520 seconds since midnight (the TIME value), and the DATETIME value was 1,406,633,520 seconds since midnight January 1, 1960.

Obviously numbers like this are difficult for us to work with (unless you have a SAS watch that tells the time in seconds since midnight) so SAS has created a number of tools for us to use to manipulate and display these values.

WHAT IS A SAS DATE AND TIME LITERAL?
One of the simplest of these tools is are literal strings. These are used when you would like to insert a constant DATE, TIME, or DATETIME value into a DATA step value. The following DATA step creates three constant values.

```
data sampdate;
sampdate = '28jul2004'd;
samptime = '11:32't;
```

```
sampdtime= '28jul2004:11:32'dt;

put sampdate=;
put samptime=;
put sampdtime=;
run;
```

The LOG shows:

```
sampdate=16280
samptime=41520
sampdtime=1406633520
```

Notice that the literal values are inclosed in quotes and immediately followed by a letter that tells SAS how to interpret the literal string. Dates must be in ddmonyyyy form, while time values are hh:mm:ss and datetime values are a combination of the two.

Although the three forms of date/time measurement are each important, they do not receive equal application. Throughout the remainder of this paper, although a given example will often only deal with only one of the measurement forms, say DATE, the reader should keep in mind that the discussion will often apply equally to the other two forms.

DATE AND TIME FUNCTIONS

A number of functions have been created to help us work with DATE, TIME, and DATETIME values. They can be used to create, modify, convert, and otherwise manipulate these values from one form to another.

CREATING DATE AND TIME VALUES

Sometimes we have date and/or time information that may not be in SAS date/time form. Functions can be used to establish date/time values. The following data step demonstrates some of the functions that can be used to create date/time values.

```
data dtvalues;
day=28;
mon=7;
yr=2004;
hr=11;
min=32;
sec=0;

sampdate = mdy(mon,day,yr);
samptime = hms(hr,min,sec);
sampdtime= dhms(sampdate,hr,min,sec);
current = today();

put sampdate=;
put samptime=;
put sampdtime=;
run;
```

The TODAY and DATE functions return the current date as stored on the computer's clock.

TAKING DATETIME VALUES APART

If you have a DATETIME value and want to create DATE and TIME values the DATEPART and TIMEPART functions can be used to convert the number of seconds since the beginning of time to days and seconds since midnight.

```
data samppart;
sampdtime= '28jul2004:11:32'dt;
sampdate = datepart(sampdtime);
samptime = timepart(sampdtime);

put sampdate=;
put samptime=;
put sampdtime=;
run;
```

You can also break up both DATE and TIME values using additional functions.

```
data allapart;
sampdate = '28jul2004'd;
day=day(sampdate);
mon=month(sampdate);
yr =year(sampdate);

samptime= '11:32't;
hr = hour(samptime);
min= minute(samptime);
sec= second(samptime);

put sampdate= ;
put day= mon= yr=;
put samptime=;
put hr= min= sec=;
run;
```

USING DATE AND TIME VALUES

Once you have created a SAS date it can be easily manipulated by a variety of other SAS functions. The following DATA step converts a SAS date into some other forms.

```
data reform;
sampdate = '28jul2004'd;

julian = juldate(sampdate);
julian7= juldate7(sampdate);
quarter= qtr(sampdate);
dayofwk= weekday(sampdate);


put sampdate= ;
put julian=;
put julian7=;
put quarter=;
put dayofwk=;
run;
```

The LOG shows:

```
sampdate=16280
julian=4210
julian7=2004210
quarter=3
dayofwk=4
```

WORKING WITH INTERVALS
It is not at all unusual to need to work with date/time intervals. Of course the number of elapsed days is a simple subtraction when dealing with dates, but what about other intervals? One common interval that must be calculated is age in years. There are several ways to calculate the number of elapsed years, however the newer function YRDIF is the most accurate. The following calculates my son's age as of July 28, 2004.

```
data age;
dob = '04jun1975'd;
age = yrdif(dob,'28jul2004'd,'act/act');

put dob=;
put age=;
run;
```

The LOG shows that he was born 5632 days after the beginning of time and is just over 29 years old (incidently he was born when I was 5).

```
dob=5632
age=29.151860169
```

The third argument allows the use of alternate month and year definitions that are sometimes used by accountants. ACT indicates that you want to use the actual number of days per month and per year. This option, however allows you to use counting intervals were all years have 360 days and all months have 30 days.

The INTNX and INTCK functions are also used to calculate intervals and are not limited to counting the number of elapsed years. Both use an argument to specify the type of date/time interval of interest. The INTCK function counts the number of intervals between two dates.

```
data ageint;
dob = '04jun1975'd;
yrs = intck('year',dob,'28jul2004'd);
months = intck('month',dob,'28jul2004'd);
weeks = intck('week',dob,'28jul2004'd);
qtrs = intck('qtr',dob,'28jul2004'd);

put yrs=;
put months=;
put weeks=;
put qtrs=;
run;
```

The LOG shows:

```
yrs=29
months=349
weeks=1521
qtrs=117
```

Notice that the calculation for the number of years is different from that generated by YRDIF. This is because the INTCK and INTNX functions base the interval from the start of the respective intervals. This means that YRS would have been 29 for any DOB in 1975 as well as for any second date in 2004. This behavior can be modified using the shift operators and alignment options shown later.

You can also advance a date/time using the INTNX function. This has proved useful for the determination of start and end points of periods of time. Suppose you need to determine the first and last date for the month that contains the current sampling date.

```
data period;
sampdate = '28jul2004'd;
start = intnx('month',sampdate,0);
stop = intnx('month',sampdate,1) - 1;

put sampdate=;
put start= ;
put stop= ;
run;
```

The third argument of the INTNX function specifies how many intervals to advance the date value. For START we advance it 0 months and since INTNX always deals with the start of the month, this results in 01jul2004 (for any date in July). When calculating the value for STOP, the INTNX function advances 1 month (01aug2004) and then we subtract one day to get the last day in July.

USING FORMATS

DISPLAYING DATE AND TIME VALUES

As has been demonstrated in all of the previous examples, when a date/time value is displayed its raw value (number of days since the beginning of time) is not very readable. SAS gets around this be supplying a number of formats and informats that have been specifically tailored for use with date/time values. There are many more of these than can be shown here and it is crucial that the informed SAS programmer have a working knowledge of the full range of these formats. The following example adds formats to the PUT statements of the previous example.

```
data period;
sampdate = '28jul2004'd;
start = intnx('month',sampdate,0);
stop = intnx('month',sampdate,1) - 1;

put sampdate= worddate18.;
put start= date9.;
put stop= mmddyy10.;
run;
```

The LOG shows:

```
sampdate=July 28, 2004
start=01JUL2004
stop=07/31/2004
```

The formats of the general form of mmddyy, yymmdd, and ddmmyy (one of which was shown in the previous example allow an extension that gives the programmer the ability to control the character that separates the month/day/year values.

```
data extensions;
sampdate = '28jul2004'd;

put '  ' sampdate= mmddyy10.;
put 'b ' sampdate= mmddyyb10.;
put 'c ' sampdate= mmddyyc10.;
put 'd ' sampdate= mmddyyd10.;
put 'n ' sampdate= mmddyyn8.;
put 'p ' sampdate= mmddyyp10.;
put 's ' sampdate= mmddyys10.;
run;
```

The LOG shows:

```
  sampdate=07/28/2004
b sampdate=07 28 2004
c sampdate=07:28:2004
d sampdate=07-28-2004
n sampdate=07282004
p sampdate=07.28.2004
s sampdate=07/28/2004
```

When using an extension of 'n' (none) a width of 10 will cause an error.

The "anydate" informats are new with SAS 9. They are designed to allow you to read in a variety of date forms including:

- DATE, DATETIME, and TIME
- DDMMYY, MMDDYY, and YYMMDD
- JULIAN, MONYY, and YYQ.

These can be especially useful when your incoming data has a mixture of date forms. The various forms of these informats include:

• ANYDTDTE.	extracts the date portion
• ANYDTDTM.	extracts the datetime portion
• ANYDTTME.	extracts the time portion

```
options datestyle=mdy;
data new;
input date anydtdte10.;
put date;
format date date9.;
datalines;
01/13/2003
13/01/2003
13jan2003
13jan03
13/01/03
01/02/03
03/02/01
run;
```

The LOG shows:

```
13JAN2003
13JAN2003
13JAN2003
13JAN2003
        .
02JAN2003
02MAR2001
```

The DATESTYLE= system option is used to help resolve ambiguous dates and can take on values such as MDY (default), DMY, YMD, etc. Notice that the date in the fifth observation is too ambiguous to resolve and is therefore set to missing.

You can of course create your own formats and the PICTURE style of format lends itself to work with SAS date/time values through the use of 'directives'. The ability to use date/time directives in the PICTURE statement is turned on with the DATATYPE= option. This option takes on the values of DATE, TIME, and DATETIME.

```
proc format;
picture moname
   other = '%b' (datatype=date);
picture monabb
   other = '%B ' (datatype=date);
run;

data picture;
sampdate = '28jul2004'd;
put sampdate=monname.;
put sampdate=monabb3.;
run;
```

The LOG shows:

```
sampdate=July
sampdate=Jul
```

There are over 15 directives and the case of the letters used as directives is important. Some of the available directives include:

Y	year
m	month
d	day
H	hour
M	minute
S	second
B	month abbreviation
b	month name

The following format creates a DATETIME string in a form that could be used by DB2.

```
proc format;
picture dbdate
   other = '%Y-%0m-%0d:%0H:%0M:%0S' (datatype=datetime);
   run;
```

```
data _null_;
sampdtime= '28jul2004:11:32'dt;
put sampdtime= dbdate.;
run;
```

The LOG shows:

```
sampdtime=2004-07-28:11:32:00
```

CREATING A SAS DATE/TIME VALUE USING THE INPUT FUNCTION
Sometimes you will receive a data table that has a date stored as a character string. In addition to using the ANYDATE informats, which were shown above, there are other informats that can be used with the INPUT function to create a SAS date/time value.

The trick is to pick the informat that matches the form of the character string. In the following example the string stored in the character variable CDATE (it's a string - NOT a SAS date) is in the ddmonyyyy form which corresponds to the DATE9 informat.

```
data chardate;
cdate = '28jul2004';
sampdate = input(cdate,date9.);

put sampdate= sampdate=date9.;
run;
```

The LOG shows that the variable SAMPDATE is indeed a SAS date value.

```
sampdate=16280 sampdate=28JUL2004
```

THE %SYSFUNC MACRO LANGUAGE FUNCTION

Most of the functions available in the DATA step are also available in the SAS macro language. You can access these functions in the macro language as if you were in the DATA step by using the %SYSFUNC macro language function. This function has two arguments. The first is the DATA step function that is to be executed and the second an optional format to apply to the result of the function.

RETRIEVING A DATE
The following macro uses %SYSFUNC to call the TODAY function to retrieve the current date (a numeric value) and then to display it as a formatted character string.

```
%macro currdate;
%trim(%left(%qsysfunc(date(),worddate18.)))
%mend currdate;

title "Data updated on: %currdate";
```

CONVERTING A CHARACTER DATE TO A SAS DATE
If you have a date string that you want to convert to a SAS date, the process is similar to what you would use in the DATA step. The macro variable &SYSDATE9 contains the date the current SAS session was started (28Jul2004). We want to create a macro variable that contains the SAS date (16,280).

```
%let numdate = %sysfunc(putn("&sysdate9"d,5.));
%put sysdate is &sysdate9;
```

```
%put numdate is &numdate;
```

The LOG shows:

```
15   %put sysdate is &sysdate9;
sysdate is 28JUL2004
16   %put numdate is &numdate;
numdate is 16280
```

SHIFT OPERATORS AND ALIGNMENT OPTIONS

As was discussed earlier, the INTNX and INTCK functions can be very useful when dealing with date/time intervals. The examples showed that these functions always specify intervals that are by default measured from the start of the interval. Also the types of intervals that are available for your use may not be sufficient for your needs. Shift operators and alignment options add a great deal of flexibility to these functions.

ALIGNMENT OPTIONS FOR INTNX

The alignment options allow the INTNX function to use either the Beginning (the default), Middle, or End of the interval. The alignment option becomes the optional fourth argument. In an earlier example that calculated the end of the month we could have instead used the following to determine the value for STOP.

```
data period;
sampdate = '28jul2004'd;
start = intnx('month',sampdate,0);
stop = intnx('month',sampdate,0,'e');

put sampdate= worddate18.;
put start= date9.;
put stop= mmddyy10.;
run;
```

The LOG shows:

```
sampdate=July 28, 2004
start=01JUL2004
stop=07/31/2004
```

USING SHIFT OPERATORS

Shift operators extend the number and types of available intervals. The topic is quite complex so consult the documentation for more details. Basically a suffix is added to the interval name. For the interval YEAR the new interval could be specified as YEARm.s; where 'm' is the multiplier and 's' is the shift.

Normally when we specify YEAR we want an interval of one year and that year is to start on January 1st. By adding a multiplier of 2, YEAR2 would specify a two year interval. Adding a shift of 3 would specify an interval start of March rather than January. Taken together YEAR2.3 would specify a two year interval with 01March as an interval start. For groups with a fiscal year starting in October the interval becomes YEAR.10 .

```
data period;
sampdate = '28jul2004'd;
yrstart = intnx('year',sampdate,1);
yrstart2 = intnx('year2',sampdate,1);
yrstart23 = intnx('year2.3',sampdate,1);

put sampdate= worddate18.;
put yrstart= date9.;
put yrstart2= date9.;
put yrstart23= date9.;
run;
```

The LOG shows:

```
sampdate=July 28, 2004
yrstart=01JAN2005
yrstart2=01JAN2006
yrstart23=01MAR2006
```

USING DATES IN SAS/GRAPH

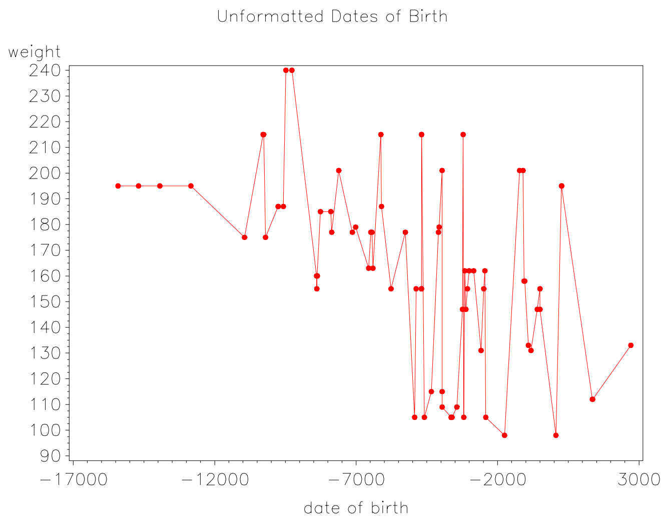
When producing plots and charts it is common to use date/time values on one or both of the axes. Since SAS dates are numeric, graphing dates without taking their nature into consideration often results in an unsatisfactory graph.

The following graphs each plot the weight of patients against their date of birth (a SAS date ranging from the years 1915 to 1970). In the first plot the Date of Birth is plotted as it is in the data.

```
proc gplot data=clinics;
plot wt*dob;
title 'Unformatted Dates of Birth';
run;
```

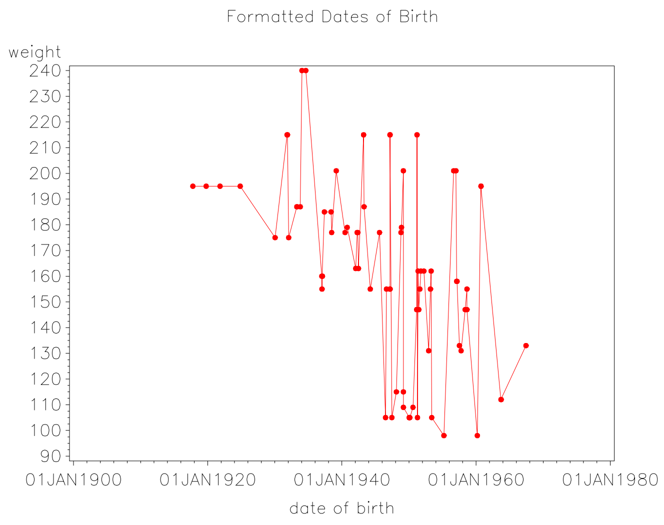
Here you can see that the date of birth (DOB) ranges from -17,000 to 3,000. The axis values are of course worthless to anyone without a pocket calender of SAS dates.

We do not want to change the values in the data since they are SAS dates, but we can easily change how they are to be displayed by using an appropriate date format.



By assigning a date format to the DOB variable the horizontal axis becomes much easier to read.

```
proc gplot data=clinics;
plot wt*dob;
title 'Formatted Dates of Birth';
format dob date9.;
run;
```

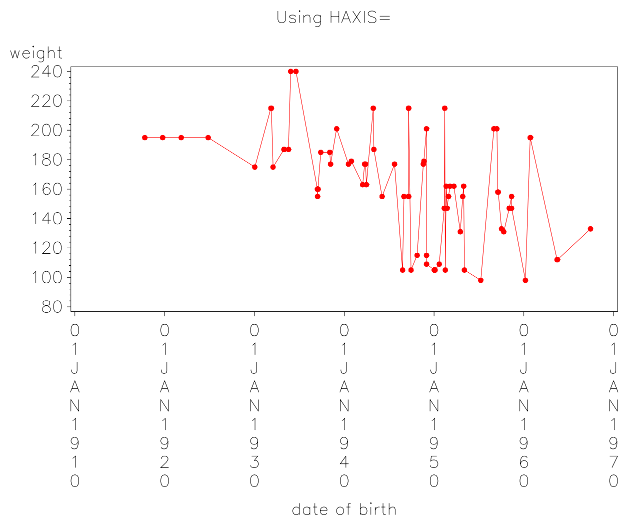


In this case SAS/GRAPH has not only recognized that these are date values, but has made an attempt to create major tick marks at appropriate date boundaries. In this case every 20 years on the first of January. It does not always make such a successful selection of intervals, and consequently we often need a bit more control.

The selection of the major tick marks can be done manually or automatically with direction from the programmer. The tick marks intervals can also be selected by using the same intervals designations that are accepted by the INTCK and INTNX functions. This includes the use of shift operators.

In the following example, plotting by decade is accomplished by using the HAXIS= option on the PLOT statement.

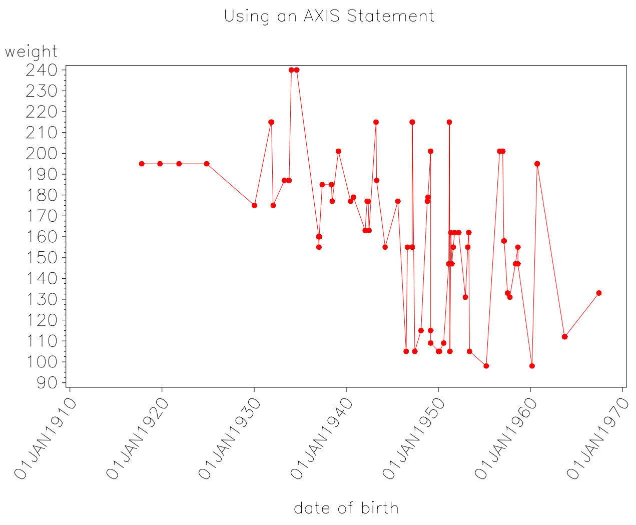
```
proc gplot data=clinics;
plot wt*dob/haxis='01jan1910'd to '01jan1970'd by year10;
title 'Using HAXIS=';
format dob date9.;
run;
```



Although the major tick marks have been selected as we requested, the axis labels are not as nice as we would have liked. Fortunately the same interval designations that we used with the HAXIS= option are also available with ORDER= option in the AXIS statement.

In the next example, the AXIS statement is defined with the ORDER= option to specify the same intervals. The VALUE= option is used to slant the tick mark labels by 55 degrees.

```
axis1 order=('01jan1910'd to '01jan1970'd by year10)
      value=(a=55 r=0);
proc gplot data=clinics;
plot wt*dob/haxis=axis1;
title 'Using an AXIS Statement';
format dob date9.;
run;
```



Although not used in this example we could have also shifted the tick marks to measure from March by using YEAR10.3 .

SYSTEM OPTIONS

In addition to the DATESTYLE= SAS9 system option used with the ANYDTE informats which was mentioned in a previous example, there are a couple of other system options that deal with dates.

The DATE system option turns on or off (NODATE) the printing of the date in the upper right corner of the output page. The date printed is the value of the &SYSDATE macro variable, which is assigned its value when the current SAS session is initialized.

The YEARCUTOFF system option was added to provide a methodology for handling two digit years. If we specify the date literal '28jul04'd, we could intend either 2004, 1904, or even 1704. The YEARCUTOFF option is used to eliminate this ambiguity. SAS uses the value of the YEARCUTOFF option to determine a 100 year interval into which all two digit years are to fall. Under Version 7 and later the default value for YEARCUTOFF is 1920 (it was 1900 in V6). This means that if you ever use a two digit year, its appropriate 4 digit year value will be selected from the 100 year period starting with 1920.

The best strategy, when you can take advantage of it, is to never specify two digit years. Use formats such as DATE9. in preference to DATE7. to show all four digits.

SUMMARY

SAS offers a wide array of tools to aid us as we work with date, time, and datetime values. These tools include a wealth of functions, formats and options, and they can be used in a variety of ways within the SAS System.

Each of the examples in this paper have just scratched the surface of the available capabilities. There is much more. Explore.

Most data tables are going to have a time component. Be sure that you can take advantage of the power of SAS to manipulate and display these values.

Know the functions. Know the formats. Know the options. Be sure that you are able to get a date when you need one!

ABOUT THE AUTHOR

Art Carpenter's publications list includes three books, and numerous papers and posters presented at SUGI and other user group conferences. Art has been using SAS® since 1976 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Professional™ and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
P.O. Box 430
Oceanside, CA 92085-0430

(760) 945-0613
art@caloxy.com
www.caloxy.com

REFERENCES

Carpenter, Art, *Carpenter's Complete Guide to the SAS® Macro Language 2nd Edition,* Cary, NC: SAS Institute Inc., 2004.

Karp, Andrew, 2004, "Time Series Magic: Using PROC EXPAND with Time Series Data", presented at the 12th Annual WUSS conference and published in its proceedings.

Karp, Andrew , 2000, "Working With SAS Date and Time Functions," *NorthEast SAS Users Group 13th Annual Conference Proceedings,* 197-203. Also presented at the 12th Annual WUSS conference and published in its proceedings (2004).

Rhoades, Mike, 2003, "Starts and Stops: Processing Episode Data With Beginning and Ending Dates" SUGI 29 Conference Proceedings, paper #260-29.

TRADEMARK INFORMATION