Why the Bell Tolls 108 Times: Stepping through Time with SAS

Peter Eberhardt, Fernwood Consulting Group Inc,
Toronto, ON Canada

For many new SAS programmers SAS date variables are often a source of confusion.

Lets look at some examples of how SAS stores date variables then how they can be displayed and manipulated.

First lets look at calendar dates

Yesterday	Today	Tomorrow
Fri Jul 6, 2012	Sat Jul 7, 2012	Sun Jul 8, 2012
2012年07月07日	2012年07月08日	2012年07月09日
Ven Jul 6, 2012	Sam Jul 7, 2012	Dim Jul 8, 2012

English , Chinese or French these are the same dates

One day separates each

What is Missing?

Yesterday	Today	Tomorrow
Fri Jul 6, AD 2012	Sat Jul 7, AD 2012	Sun Jul 8, AD 2012
CE 2012年07月07日	CE 2012年07月08日	CE 2012年07月09日
Ven Jul 6, AD 2012	Sam Jul 7, AD 2012	Dim Jul 8, AD 2012

AD (Anno Domini) or CE (Common Era) provides the reference point for the dates

Our dates have a "starting" point from which they uniformly move forward. In Western society this is annotated with AD (Anno Domini). In practice the AD is rarely added to dates although it is implied.

Our base time is Jan 1 AD 1.
The day before is Dec 31, 1 BC (Before Christ).
The AD goes before the year whereas the BC goes after.

Thursday 4 October AD 1582
Was followed by
Friday 15 October AD 1582

1582
October

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	15	16
17	18	19	20	21	22	23

Pope Gregory XIII in the Papal bull
Inter gravissimas
reformed the calendar giving us what is known as the Gregorian Calendar.

This replaced the Julian Calendar which had been in effect since 45 BC.

Why??
Easter could not be properly determined.

SAS Dates

We can enter a date, datetime or a time value using SAS contstants. The following code snippets show constants for dates, datetimes and times respectively

```
1    data _null_;
2         toDay = "7jul2012"d;
3         put toDay= /
4             toDay= yymmdd10. /
5             toDay= worddate23./
6             toDay= nldate23.
7    ;
8    run;

toDay=19181
toDay=2012-07-07
toDay=July 7, 2012
toDay=2012年07月07日
```

Date Constant
Date Display
Date constants are in the form "ddMMMyyyy"d
The month is not case sensitive
The variable toDay will be displayed four times in the log, each with a different format
Note that we have four different displays yet we have only ONE variable.

- **7 Jul 2012**
  - The underlying data representation (**19181**) is independant of its visual representation (**2012-07-07, July7, 2012**, etc).
  - **What is 19181?**
    - The number of days from January 1, 1960

1960-01-01 == 0	1960-01-01 == 0
1960-01-02 == 1	1959-12-31 == -1
1960-01-03 == 2	1959-12-30 == -2
…	…
2012-07-07 == 19181	1592-10-15 == -137774
	1592-10-04 == -137785

SAS dates are simply numbers. This makes it easy to add or subtract days . To get tommorow's date we only have to add 1 to today's date..

SAS Datetimes

```
8   data _null_;
9     toDay = "07jul2012:9:30:0"dt;
10    put toDay= /
11        toDay= datetime. /
12        toDay= nldatm. /
13        toDay= datetime18.3 /
14        toDay= nldatm18. ;
15  run;

toDay=1657272600
toDay=07JUL12:09:30:00
toDay=2012年07月07日 09时30分00秒
toDay=07JUL12:09:30:00.0
toDay=07JUL12:09:30:00
```

DateTime Constant
DateTime Display
Date constants are in the form "ddMMMyyyy:hh:mm:ss"dt
The month is not case sensitive
The variable toDay will be displayed five times in the log, each with a different format
Note that we have five different displays yet we have only ONE variable.

- **7 Jul 2012 @ 9:30 am**
  - **What is 1657272600?**
    - The number of seconds since January 1, 1960 at midnight
    - Datetime variables are floating point numbers (can contain partial seconds)

SAS Times

```
16  data _null_;
17    morning = "9:30"t;
18    put morning= /
19        morning= time. /
20        morning= timeampm. /;
21  run;

morning=34200
morning=9:30:00
morning=9:30:00 AM
```

Time Constant
Time Display
Date constants are in the form "hh:mm:ss"t
Times after noon can be specified using a 24 hour clock or by addin gPM
Note that we have three different displays yet we have only ONE variable.

- 9:30
  - **What is 34200?**
    - The number of seconds since midnight
    - Time variables are floating point numbers (can contain partial seconds)

SAS Formats

The underlying data representation
19181
1657272600
34200
is independant of its visual representation
2012-07-07 , July 7, 2012
7JUL12:09:30:00, 2012年07月07日 09时30分00秒
9:30:00, 9:30:00 AM

SAS Formats convert the data representation into the visual representation

SAS Informats

Informats are the "mirror image" of formats
Turn a visual representation into the underlying data representation

```
data _null_;
   input toDay      yymmdd10. +1
         tomorrow   date9.    +1
         toDayDT    datetime16. +1
         morning    time5.;
   put toDay= tomorrow= toDayDT= morning=;
datalines;
2012-07-07 08jul2012 07DEC12:09:30:00 09:30
2012-07-08 09jul2012 07DEC12:19:30:00 21:30
;;
run;
toDay=19181 tomorrow=19182 toDayDT=1670491800 morning=34200
toDay=19182 tomorrow=19183 toDayDT=1670527800 morning=77400
```

Using Informats To read data
Display with default format
Input data with different visual representations
Output

SAS Informats convert the visual representation into the data representation

SAS Functions

SAS provides numerous functions to manipulate date, datetime, and time time variable. Following are some of the more common.

date()/today()
- Returns the current date
- Useful for capturing the date for titles or footnotes

```
%let titledate = %sysfunc(today(),yymmdd10.);
footnote1 "Produced on &titledate";
```

The %sysfunc() macro will call the function today() and apply the YYMMDD10. format to the result

time()
- Returns the current time

```
%let titledate = %sysfunc(date(),date9.);
%let titletime = %sysfunc(time(), timeampm.);
footnote1 "Produced on &titledate at &titletime";
```

The macro variables are available for titles and footnotes

datepart()
- Returns the date part of a datetime variable
- Useful when dealing with dates from DBMS's which often appears as datetime

```
data fromSQLServer;
     set sql.dataTable;
     sasDate = datepart(sqlDate);
     drop sqlDate;
     format sasDate yymmmdd10.;
run;
```

MS SQL Server returns dates as SAS datetime variables. Use **datepart()** to create a SAS Date

timepart()
- Returns the time part of a datetime variable

```
data fromSQLServer;
     set sql.dataTable;
     sasTime = timepart(sqlDate);
     drop sqlDate;
     format sasTime timeAMPM.;
run;
```

SAS Functions, continued

day(dateVar)
- Returns the day (1 – 31) of a SAS date variable

month(dateVar)
- Returns the month (1 – 12) of a SAS date variable

year(dateVar)
- Returns the year of a SAS date variable

qtr(dateVar)
- Returns the quarter (1 – 4) of a SAS date variable

mdy(mm, dd, yy)
- Returns a SAS date from the three integer variables for month (mm), day (dd), and year (yy) input

hms(hh, mm, ss)
- Returns a SAS time variable for the three variables hour (hh), minute (mm), and second (ss). The seconds input can be floating point

dhms(date, hh, mm, ss)
- Returns a SAS date time variable from the four variables for date, hour, minute, and second

intck()
- Returns the count of the number of interval boundaries between two dates, two times, or two datetime values

intnx()
- Increments a date, time, or datetime value by a given time interval, and returns a date, time, or datetime value.

To use intck() and intnx() properly you need to understand boundaries and intervals

intck()/intnx Intervals

Interval
- Represents the measurement period
- Character value
- Common values are (not case sensitive):
  - 'day'
  - 'week'
  - 'month'
  - 'year'
  - 'qtr'

intck()/intnx Boundaries

Boundary
- Starting point of each interval

Interval	Boundary
'day'	midnight
'week'	sunday
'month'	first day of month
'year'	Jan 1
'qtr'	Jan 1, Apr 1, Jul 1, Oct 1

intck()

Intck('interval', start, end)
- **start** can be date, time, or datetime
- **end** must be same as start

```
33   data _null_;
34        start = '01jan2012'd;
35        end = '01jul2012'd;
36        days1 = end - start;
37        days2 = intck('day', start, end);
38        weeks = intck('WEEK', start, end);
39        months = intck('month', start, end);
40        year = intck('year', start, end);
41        put days1= days2= weeks= months= year=;
42        end = '30jun2012'd;
43        days1 = end - start;
44        days2 = intck('day', start, end);
45        weeks = intck('WEEK', start, end);
46        months = intck('month', start, end);
47        year = intck('year', start, end);
48        put days1= days2=      = months= year=;
49   run;

days1=182 days2=182 weeks=26 months=6 year=0
days1=181 days2=181 weeks=25 months=5 year=0
```

Period from Jan 1, 2012 to July 1, 2012
Calculate some intervals
Display results in LOG
Period from Jan 1, 2012 to June 30, 2012
Note the difference in weeks and months yet only one day difference

Where does the time go?

```
50   data _null_;
51        start = '31dec2011'd;
52        end = '01jan2012'd;
53        days1 = end - start;
54        days2 = intck('day',      , end);
55        weeks = intck('WEEK', start, end);
56        months = intck('month', start, end);
57        year = intck('year', start, end);
58        put days1= days2= weeks= months= year=;
59   run;

days1=1 days2=1 weeks=1 months=1 year=1
```

Period from Dec 31, 2011 To Jan 1, 2012
One day, one year and one month elapsed

intnx()

intnx('interval', start, increment <, *'align'*>)
is the number of **intervals** to add to **start**
can be negative

- **align**: controls position of the resultant date within the interval. Optional
  - 'beginning' ('b')
    - This is the default
  - 'middle' ('m')
  - 'end' ('e')
  - 'same' ('s')

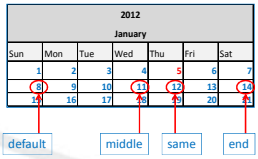
Why the Bell Tolls 108 Times: Stepping through Time with SAS

intnx()

```
50   data _null_;
51       start = '05jan2012'd;
52       weeks = intnx('week', start, 1);
53       weeksE = intnx('week', start, 1, 'e');
54       weeksS = intnx('week', start, 1, 's');
55       weeksM = intnx('week', start, 1, 'M');
56       put weeks= yymmdd10. weeksE= yymmdd10. weeksS= yymmdd10. weeksM= yymmdd10.;
57   run;

weeks=2012-01-08 weeksE=2012-01-14 weeksS=2012-01-12 weeksM=2012-01-11
```

default (beginning)
end
same
middle



Adding one week to Jan 5 can result in four different end values

End of the Month

```
58   data _null_;
59       start = '11jan2012'd;
60       monthStart = intnx('month', start, 0, 'b');
61       monthEnd   = intnx('month', start, 0, 'e');
62       put monthStart= yymmdd10. monthEnd= yymmdd10.;
63   run;

monthStart=2012-01-01 monthEnd=2012-01-31
```

Add 0 (zero) months, set to beginning
Add 0 (zero) months, set to end

There and Back Again

The underlying data representation of date, datetime and time variables

19181
1657272600
34200

is independant of its visual representation

2012-07-07 , July 7, 2012
7JUL12:09:30:00, 2012年07月07日 09时30分00秒
9:30:00, 9:30:00 AM

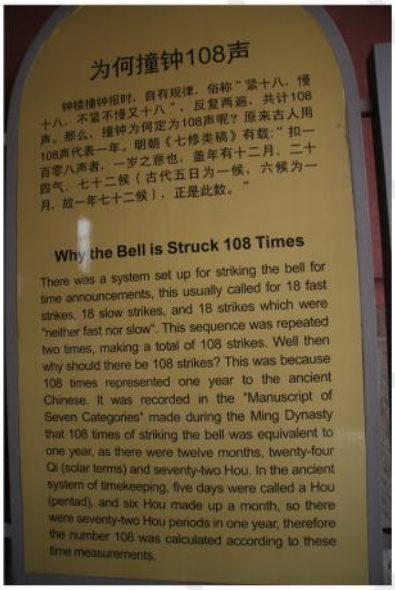
SAS Formats convert the data representation into the visual representation

SAS Informats convert the visual representation into the data representation

SAS functions are available to manage virtually all aspects of date manipulation

Understand **Intervals** and **Boundaries** when manipulating SAS dates

Why Does the Bell Toll 108 Times?



Peter Eberhardt
Fernwood Consulting Group Inc.
Toronto, ON Canada
peter@fernwood.ca

Time is the thing that keeps everything from happening at once