technical tips

Accurately calculating age
in only one line

A frequent need of SAS software users is to determine a person's age, based on a given date and the person's birthday. Although no simple arithmetic expression can flawlessly return the age according to common usage, efficient code to do so can be written using SAS software's date functions. This article, by SAS software user William Kreuter, presents a one-line solution for this purpose.

Put SAS date functions to work for you

Many kinds of work require the calculation of elapsed anniversaries. The most obvious application is finding a person's age on a given date. Others might include finding the number of years since any event has occurred, such as an index date for medical treatment or the start of a magazine subscription.

However, because of the way the modern Gregorian calendar is constructed, there is no straightforward arithmetic method that produces a person's age, stated according to common usage — common usage meaning that a person's age should always be an integer that increases exactly on a birthday. (Persons born on a February 29 are a special case that will be addressed later.) It is often important to compute an age that exactly conforms to this usage, for example so that the data will be consistent with the age written on a medical record.

Exact computation of ages and elapsed anniversaries must take into account all rules governing lengths of months and lengths of years. While the rich set of functions and programming constructions available in the SAS DATA step language allows many possible solutions, this article presents a concise solution that relies on letting the SAS date functions do all the work.

What doesn't work

Often, SAS software users attempt to compute age using an expression such as:

```
age = (somedate - birth) / 365.25;
```

where **somedate** and **birth** are SAS date variables (or constants or expressions). Clearly this usually doesn't return an integer and therefore it is not stating an age according to colloquial usage. That problem can be addressed by:

```
age = floor( (somedate - birth) / 365.25);
```

Now we're at least getting integers. In fact, for most dates in a given year this statement does produce the correct result. But in most years, age will increment on the wrong day. To account for the Gregorian calendar's idiosyncrasies, some users make attempts such as:

```
age = floor( (somedate - birth) / 365.2422);
```

However, extending the denominator to any number of significant digits doesn't help. Astronomers define several kinds of "years" for various technical uses, but the Gregorian calendar uses a different concept of "year" in which there are always either 365 or 366 days. No algorithm of this kind perfectly models such an interval.

The Julian calendar, which was introduced in 46 BC, gave every fourth year 366 days. Because this slowly causes a discrepancy between the calendar and the seasons, Pope Gregory XIII proclaimed the Gregorian calendar in 1582. The new rule provided that every fourth year will have 366 days, except for years divisible by 100 but not 400. Thus the year 2000 will be a leap year, but 2100 will not.

How SAS date functions can help

As we've seen, the Gregorian calendar, and hence an integer count incremented on an anniversary date, cannot be modeled with simple arithmetic. A completely accurate

approach requires coding all of the rules for which years are leap years and all the rules for the number of days in each month.

This is where SAS software's date functions help. Date functions such as **intck** and **intnx** have the needed rules built in. Particularly, **intck** returns intervals that correctly account for the idiosyncrasies of the Gregorian calendar. However, a little tweaking is necessary to get exactly what we need.

Because **intck** alone won't produce the number of years between successive anniversaries given an arbitrary birth date or starting date, a tweak is needed to find how old the person is in months. Then, simple arithmetic will turn this number into what we need — years that always increment on the anniversary date.

Again, consistent with common usage, we want the number of months always to be an integer and we want it to increment exactly on the same day each month (or on the first day following a month that is too short for the same day to occur). Generally, the expression

```
intck('month',birth,somedate)
```

returns the number of times the first day of a month is passed between **birth** and **somedate**. An enhancement is needed to alter this into the number of times the same day of the starting month is passed. This simply consists of subtracting one month if the day number of **somedate** is earlier than the day number of **birth**. Although one could program this concept using a separate if-then statement, it can be calculated more concisely as a logical expression returning a 0 or 1 value. The 0 or 1 is then subtracted from the result of intck, as in the following example.

```
intck('month',birth,somedate)
      - (day(someday) < day(birth))
```

This now gives exactly the correct number of months for any pair of dates.

A one-line solution

Converting months to years, we get:

```
age = floor
     ((intck('month',birth,somedate)
      - (day(someday) < day(birth))) / 12);
```

This can be conveniently set up as a macro:

```
%macro age(date,birth);
floor ((intck('month',&birth,&date)
        - (day(&date) < day(&birth))) / 12)
%mend age;
```

This is an example of how the macro is used in a SAS DATA step:

```
age = %age(somedate,birth);
```

For example, the following lines:

```
age = %age('28aug1998'd,'24mar1955'd);
put age=;
```

will cause the following message to be placed on the log:

```
AGE=43
```

When this won't work

There are only two instances where this approach might fail to yield the expected result.

1. The birthday is February 29, and during non-leap years the person celebrates the birthday on February 28. The solution described here would treat the birthday during non-leap years as March 1. In a random population this should affect at most one out of 1,461 persons, or less than 0.07 percent of the population. If desired, extra lines of code can accommodate this situation.

2. A person's age is to be calculated at a time in history when, in some particular country, the Gregorian calendar was not in use. Beginning with the earliest date that is valid in SAS software — January 1, 1582 — SAS software uses the Gregorian calendar. That is the year that France, Italy, Luxembourg, Portugal, and Spain replaced the Julian calendar with the Gregorian. (The Gregorian calendar was first implemented so that the day after October 4, 1582 was October 15, 1582. Nevertheless, SAS software recognizes 31 days in the month of October, 1582.) While the rest of Roman Catholic Europe switched shortly after1582, the United Kingdom and its colonies did not move to the Gregorian calendar until 1752. Many other countries switched even later, including the Soviet Union in 1918 and Greece in 1923. Some historic dates therefore might be handled in a misleading manner — a problem which, it should be noted, is true of any use of SAS dates in such instances. Nevertheless, given likely scenarios, age will be computed correctly in every country and era.

This tip was written by William Kreuter, a senior computer specialist at the University of Washington in Seattle. He has used SAS software in public health research since 1981, and now specializes in manipulating large data sets for the School of Public Health and the School of Medicine. He can be reached at billyk@u.washington.edu.

If you have a SAS technical tip you'd like to share, send it to us at editor@sas.com.

I have some questions about SAS software, OLE, and Visual Basic. I have heard that you can use Visual Basic to write a front-end application and use OLE to request SAS processing in the background. Is this correct? If so, how is this done and where can I get more information?

In the Windows environment, you can take advantage of OLE automation, which is the Windows mechanism for manipulating an application's objects from outside that application.

Since Release 6.11, SAS software has been able to perform as an OLE automation server on Windows 95 and Windows NT. This means that other applications can programmatically control the SAS System.

For instance, an Excel or Visual Basic program can create a SAS session and control it using the methods and properties that SAS software makes available.

SAS software provides a variety of methods and properties to manage code submission from the PROGRAM EDITOR window and the execution of commands. By default the SAS session is hidden, so other applications can access SAS functionality without using the SAS GUI. The front-end tool can be any application that can act as an OLE automation controller (such as VB).

Release 6.12 online help provides detailed documentation for using SAS software as an OLE automation server. From the menu bar, select Help -> SAS companion -> What's New in Release 6.12 for Windows -> Controlling the SAS System using OLE Automation.

Can SAS software be embedded in a Visual Basic application? If so, how?

It depends on what you mean. SAS functionality can be accessed through a Visual Basic front end (using techniques such as automation, see above), but SAS objects, such as SAS/EIS objects or SAS/AF widgets, cannot be embedded in a Visual Basic application. You can use automation to script SAS software (control it programmatically) from Visual Basic, but SAS software's visual objects cannot be dropped into another software application.

How does SAS software access data stored in Lotus Notes?

SAS/ACCESS Interface to ODBC provides access to any ODBC-compliant database package for which you have an ODBC driver installed. This interface allows users to retrieve information about existing Notes documents in a Lotus Notes database, provided a Notes ODBC driver and data source are configured.

In addition, the NOTESBDB access method lets you add new Notes documents to a Lotus Notes database. You can use this access method to automatically output SAS data to Lotus Notes. You can build an interactive SAS/AF application that populates a Notes database, or you can redirect the output of a batch program to a Notes database.

These and other details about SAS software's integration with Lotus Notes are available from SAS Institute's Web site at www.sas.com//partners/enterprise/msinfo/whitepapers.htm

DOWNLOAD IT!

NEW UTILITY CONVERTS SAS/AF® FILES TO HTML

For organizations with SAS/AF applications that use CBT entries (for example, help system information tied to a SAS application), a utility is now available for easily converting CBT files to HTML for viewing with Web browsers. Customers can use this tool as a starting point for converting information to HTML while preserving the structure and linking information in the original CBT files.

The CBT2HTML utility is an SCL application with a SAS/AF interface. Users supply the name of an input library and catalog as well as a target directory for the output HTML files. The utility will maintain the appearance of the input files by preserving line breaks and by writing every converted frame to a separate file in the output directory. Links are preserved as HTML links, and the user can page through entries by using the Forward and Back buttons in the browser.

This utility is available through the DEMOS & DOWNLOADS -> SAS/AF section on the SAS Institute World Wide Web site, www.sas.com. Check this location for system requirements, download instructions, and more detailed information on this experimental utility.