Paper 19-2010

Happy Birthday! But… How Old Are You *Really*?

Elizabeth Axelrod, Abt Associates Inc., Cambridge, MA

Abstract

Chances are that at some point in your SAS® career you've had to calculate age. There are various ways to do this, but one of the most common ones does not always produce a correct value. This paper presents several alternatives to calculate age, and illustrates what works most accurately - and what doesn't.

Introduction

By conventional reckoning, our age increases by one each year on our birthday. It's not a smooth transition during the year from 20 to 21. You're 20 for a whole year, then boom – you're 21! Also, our age increases by one year on our birthday regardless of how many days the previous year contains (365 or 366). These two factors make age interesting to calculate.

SAS and dates

SAS stores dates as consecutive integers starting with January 1, 1960, as 0 (zero). This enables us to use SAS built-in functions and arithmetic logic to manipulate dates in a variety of ways, including determining intervals between dates. To calculate age, we need two points in time: Date of Birth (DOB), and an AGE as OF date, which I'll call NOW. In this paper I will discuss three different formulae to calculate age, using DOB and NOW.

Method #1

The first formula that many of us learn is rather appealing in its simplicity:

```
AGE = int((NOW - DOB)/365.25);
```

First determine the number of days from DOB to NOW, then divide by the number of days in a year. Because we have leap years, we use 365.25 as an average number of days in a year. Since AGE is always a whole number, we then use the INT (or FLOOR) function to grab just the integer portion of the result. This generally produces an accurate value for age, making it a nice solution for most applications.

But as we can see in the data below, sometimes it goes wrong. As much as we might want to remain 29 forever, we really do turn 30 on our 30th birthday, and we might want a formula that produces an accurate result.

DOB	NOW	Correct Age	Age Method #1
9/10/1980	9/9/2010	29	29
9/10/1980	9/10/2010	30	29

Leap Year is the obvious pesky culprit – if all years had 365 days, then we could just divide by 365, and the formula would always produce a correct value for age. Later in this paper we'll see more examples of exactly where results go amiss (it's actually a rather interesting pattern). But for now, let's look at two other formulae that do produce accurate values for AGE.

Method #2

We have seen that using the number of days between two dates to calculate the number of elapsed years is problematic because years can have a different number of days. Here's a solution that relies on the number of *months* between two dates.

```
age = INT((INTCK("month",dob,now) - (DAY(dob) > DAY(now)))/12);
```

Essentially, this formula determines the number of months between DOB and NOW, decides whether to subtract 1, and then divides by 12 (the number of months in a year) to get number of years.

So, let’s break it down:

$$\texttt{age = }\underbrace{\texttt{INT(}}_{5}\underbrace{\texttt{(INTCK("month",dob,now)}}_{1}\ \underset{\substack{\uparrow\\3}}{\texttt{-}}\ \underbrace{\texttt{(DAY(dob) > DAY(now))}}_{2}\texttt{)}\underbrace{\texttt{/12);}}_{4}$$

1) `INTCK("month",dob,now)`

The INTCK function returns the number of units (in this case ‘months’) between DOB and NOW, including the month that’s NOW. (It’s equivalent to saying MONTHx – MONTHy + 1.) The INTCK *includes* that last month, but we have to decide if we need to exclude that month or not. That’s what the next part accomplishes.

2) `(DAY(dob) > DAY(now))`

This is a Boolean expression that returns a 1 or 0. DAY() is a function that returns the numerical day of the month (1 through 31) of a date. Is the DAY of DOB (e.g. 10) greater than the DAY of NOW (e.g. 9)? Yes (True) returns 1; No (False) returns 0.

3) We now subtract the 1 or 0 from the number of months returned from the INTCK function.

4) Divide this number by 12, then

5) Take the integer portion of the result.

Here we see that AGE is accurately calculated for the case where Method #1 failed:

DOB	NOW	Correct Age	Age Method #2
9/10/1980	9/9/2010	29	29
9/10/1980	9/10/2010	30	30

Method #3

This next formula may look a bit complex, but, in fact, it’s the most intuitive expression to determine the number of years between DOB and NOW.

```
age = YEAR(now) - YEAR(dob) –
      ( (MONTH(now) < MONTH(dob) ) or
      (MONTH(dob) = MONTH(now) & DAY(now) < DAY(dob))
 );
```

It actually works the way we think about AGE. As in Method #2, where we had to decide whether or not to subtract a month, here we have to decide whether or not to subtract a year. This is accomplished by comparing the month and day of NOW to the month and day of DOB.

Breaking it down:

$$\texttt{age = }\underbrace{\texttt{YEAR(now) - YEAR(dob)}}_{1}\ \underset{\substack{\uparrow\\3}}{\texttt{–}}$$

$$2\left\{\begin{array}{l}\underbrace{\texttt{((MONTH(now) < MONTH(dob))}}_{2.a}\texttt{ or}\\[2ex]\quad\underbrace{\texttt{(MONTH(dob) = MONTH(now) \& DAY(now) < DAY(dob))}}_{2.b}\end{array}\right.$$

```
      );
```

1) `YEAR(now) - YEAR(dob)`

Subtract the year of DOB from the year of NOW.

2) The whole purpose of all the rest of the formula is to determine whether or not to subtract 1 from the result. It's a series of Boolean expressions that ultimately will result in a 0 or 1. If either 2.a or 2.b is true, then the result is 1, and this will be subtracted from the number of years. If neither 2.a nor 2.b is true, the result is 0 (and nothing is subtracted from the number of years).

2.a) `( (MONTH(now) < MONTH(dob) )`

If the month of NOW is prior to the month of DOB, the result of the whole expression is 1.

2.b) `( MONTH(dob) = MONTH(now) & DAY(now) < DAY(dob) )`

If both the month of DOB equals the month of NOW *and* the day of NOW is prior to the day of DOB, the result of the expression is 1.

3) Now subtract 0 or 1 from the number of years.

The appeal of this method is that it relies more on the kind of logic we use when we look at two dates and try in our mind to determine age. And… like Method #2, it produces accurate results.

More Methods?

Of course! The formulae described thus far are by no means an exhaustive list. They illustrate some basic concepts, but you can mix and match functions and expressions to come up with lots of alternate solutions.

Show me the data!

Now that we have formulae to play with, let's look at some data. I wrote a little code to 'exercise' our three methods, and compare the results. The code itself (included at the end of this paper) is not particularly interesting, but it obediently churns through lots of dates to compare the methods. Since leap years are our prime suspect, I want to see how these methods behave when DOB is in a leap year, and when it's not. For this example, I use 4 DOBs:

9/10/1980 (leap year)
9/10/1981
9/10/1982
9/10/1983

I then run each of these DOBs through all 3 age-calculation methods, and for each method, I increase NOW by 1 day, 10,000 times.[1] We know that Methods 2 and 3 yield accurate values for AGE, so we are looking for cases when Method 1 returns a different value from the other methods. These are listed in the following table:

[1] Alternatively, the code can be easily modified to use 4 different values for NOW, and cycle through many different values for DOB. The results are similar.

```
                                            AGE
    DOB          NOW       Method #1  Method #2  Method #3
 09/10/1980  09/10/2010       29         30         30
 09/10/1980  09/10/2011       30         31         31
 09/10/1980  09/10/2013       32         33         33
 09/10/1980  09/10/2014       33         34         34
 09/10/1980  09/10/2015       34         35         35
 09/10/1980  09/10/2017       36         37         37
 09/10/1980  09/10/2018       37         38         38
 09/10/1980  09/10/2019       38         39         39
 09/10/1980  09/10/2021       40         41         41
 09/10/1980  09/10/2022       41         42         42
 09/10/1980  09/10/2023       42         43         43
 09/10/1980  09/10/2025       44         45         45
 09/10/1980  09/10/2026       45         46         46
 09/10/1980  09/10/2027       46         47         47
 09/10/1980  09/10/2029       48         49         49
 09/10/1980  09/10/2030       49         50         50
 09/10/1980  09/10/2031       50         51         51
 09/10/1980  09/10/2033       52         53         53
 09/10/1980  09/10/2034       53         54         54
 09/10/1980  09/10/2035       54         55         55
 09/10/1980  09/10/2037       56         57         57

 09/10/1981  09/10/2010       28         29         29
 09/10/1981  09/10/2011       29         30         30
 09/10/1981  09/10/2014       32         33         33
 09/10/1981  09/10/2015       33         34         34
 09/10/1981  09/10/2018       36         37         37
 09/10/1981  09/10/2019       37         38         38
 09/10/1981  09/10/2022       40         41         41
 09/10/1981  09/10/2023       41         42         42
 09/10/1981  09/10/2026       44         45         45
 09/10/1981  09/10/2027       45         46         46
 09/10/1981  09/10/2030       48         49         49
 09/10/1981  09/10/2031       49         50         50
 09/10/1981  09/10/2034       52         53         53
 09/10/1981  09/10/2035       53         54         54

 09/10/1982  09/10/2011       28         29         29
 09/10/1982  09/10/2015       32         33         33
 09/10/1982  09/10/2019       36         37         37
 09/10/1982  09/10/2023       40         41         41
 09/10/1982  09/10/2027       44         45         45
 09/10/1982  09/10/2031       48         49         49
 09/10/1982  09/10/2035       52         53         53

 09/10/1983  (No wrong values produced by method #1)
```

When do things go wrong? Several things are apparent:

a) Method 1 does not always calculate age accurately.
b) The only time Method 1 produces wrong results is when NOW is on a birthday.
c) Wrong results are a function of where DOB falls relative to leap years.

It is beyond the scope of this paper to delve further into the mathematics of these results. The important thing to note is that Method 1 can produce incorrect results, and these incorrect results follow a pattern.

I know what you're about to ask…

Doesn't SAS already have a built-in function to calculate AGE? Isn't that what the function YRDIF does? SAS documentation explains that the YRDIF function "returns the difference in years between two dates." [2] Doesn't that

[2] SAS Institute Inc (2010), "*SAS 9.2 Language Reference: Dictionary, Third Edition*," Cary, NC: SAS Institute Inc.

calculate age? No! The YRDIF function does not always produce an accurate value for age, nor was it intended to. References and recommended reading at the end of this paper provide more information on this topic. But we can quickly prove to ourselves that it doesn't always work by running some dates through our code again, this time adding a fourth method to calculate AGE using YRDIF.

```
       age = int(yrdif(dob,now, 'act/act'));
```

Should You Care?

Maybe… maybe not! Method 1 is only wrong in very specific circumstances. It all depends, of course, on your data, and your specific application. When Method 1 was first taught to me, I was warned that it provided an approximate value for age – close enough for the work I was doing. I've used it for many years with no ill effects, so I never gave it much thought. It was only when my results didn't exactly match some data that I was working with, that I began to explore this further.

If you need an absolutely correct value for age, then you should avoid Method 1 [AGE = int((NOW - DOB)/365.25)], or any other method that relies strictly on number of days between your two points in time.

References

SAS Institute Inc (2010), "SAS 9.2 Language Reference: Dictionary, Third Edition," Cary, NC: SAS Institute Inc.

Acknowledgements

The author wishes to thank Michael Murphy and Steven Strang for their editorial assistance, and Jack Shoemaker who has patiently provided me with his favorite age algorithm numerous times.

Recommended Reading

Delwiche, Lora D., and Slaughter, Susan J. (2010), "Computing Ages in SAS; Removal of the YRDIF Function from The Little SAS Book: A Primer, Fourth Edition", http://support.sas.com/publishing/authors/extras/61860_update.pdf

Kreuter, William (1998), "Calculating Age with Only One Line of Code", SAS Communications, 1998 and http://support.sas.com/kb/24/808.html

Shoemaker, Jack (1997), "How Old Are You?", Proceedings of the 1997 NorthEast SAS Users Group

Contact Information

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

Elizabeth Axelrod
Abt Associates Inc.
55 Wheeler Street
Cambridge, MA 02138
elizabeth_axelrod@abtassoc.com

Appendix

```
data _null_;
    file print;
    format dob now mmddyy10.;

    month=9; day=10;
    do year=1980 to 1983;
        dob=mdy(month,day,year);
        now=today();
        put;
        put 'NEW DOB: ' dob=;
        put;
        do i=1 TO 10000;
               now=now+1;
               age1 = int((now-dob)/365.25);

               age2 = INT((INTCK("month",dob,now) -
                       (DAY(dob) > DAY(now)))/12);

               age3 = YEAR(now)-YEAR(dob) -
                         ( (MONTH(now) < MONTH(dob)) or

                          (MONTH(dob) = MONTH(now) &
                           DAY(now) < DAY(dob)));

                if (age1 ne age2) or
                   (age2 ne age3) then
                         put now= age1= age2= age3=;
          end;
    end;
run;
```