More Than One Way To Age A Cat
Lloyd Butler, Health Dialog, Portland, ME

ABSTRACT
There are different ways to determine someone’s age. In every day life, we just want to know how many years someone has been alive. For example, I'm 29 until my birthday, when I suddenly become 30. But sometimes we want to know someone's age at a finer grain. Have you reached your 1/2 birthday yet? You get penalized for withdrawing from your IRA before you turn 59 1/2. What age will you be at the end of the year? To determine if your child is your dependent for tax purposes, you need to know whether they turned 19 by the end of the tax year. How do we account for leap years? If you are born on February 29th, do you age 1/4 as fast as other people? When is your birthday on non-leap years?

This paper will illustrate some of the SAS® tools to help answer questions about someone's age.

INTRODUCTION
There are many ways to interpret age. Even for a given interpretation, there are many ways to calculate age, giving either identical or slightly different results. This paper describes a few ways to calculate age, and describes the benefits of and issues with each.

The formulas presented are single lines of code intended to be used in a DATA step. It is assumed that there are two DATA step variables: dob (date of birth) and as_of_date (the day you want to know how old someone is).

SAS TOOLS
SAS has a built-in function, INTCK(), that checks the number of intervals between two dates. An interval can be a year, a month, a week, a quarter, etc. Intervals can start in a natural way, such as January for years. They can also start in non-standard ways, such as September to work with school years.

The INTCK() function is helpful, but is not sufficient to solve the age problem. This is because it only counts the number of times an interval boundary is crossed. It does not account for how close you are to the next boundary (i.e. fractional counting). Without modification, it can not account for birthdays, just birth-months.

SAS has another function, YRDIF(), that appears will give the proper result without any modifications. The function even has a parameter whose documentation states a clever technique in calculating the number of years between two dates: it can detect when a leap year occurs and will account for it.

\[
\text{yrdif(dob, as\_of\_date, 'ACT/ACT')}
\]

Unfortunately, YRDIF() is not always accurate when either the start or end year is a leap year. It fails to account for the number of days in the start and end months, and therefore either under- or over-estimates the effect of the leap day. The SAS Knowledge Base states that this inaccuracy is by design, because it is the industry standard.

EXAMPLE 1
Sally is born on 2/28/2007. On her 1st birthday, YRDIF() says she is 0.9996 years old. However, on her 2nd and 3rd birthdays, YRDIF() is precisely correct.

EXAMPLE 2
Fred is born on 5/29/2008. On his 1st birthday, YRDIF() says he is 0.99838 years old. Similarly, on his 2nd birthday, his age is given as 1.99838. Since his 4th birthday is in a leap year, as is his year of birth, YRDIF() accurately gives his age as 4.

Even though these SAS tools do not solve this problem by themselves, they can be used with other techniques to solve our specific problem.

INTEGER AGES
It is widely accepted that the following formula, attributed to William Kreuter, is the proper one when precisely calculating age in terms of integer years.

\[
\text{int}((\text{intck('MONTH', dob, as\_of\_date)} - (\text{day(as\_of\_date) < day(dob))) / 12)
\]

For people born on February 29th, Kreuter's formula assumes their birthday in non-leap years is March 1st. Along with the INTCK() function described above, it uses the DAY() function to get the day of a given date, and the INT() function to take the integer part of a number.
We can modify the `YRDIF()` function to make it work for our problem. From the examples above involving Sally and Fred, it is clear that we must add some value to the age, then take the integer part. One day is 0.0027 of a year. So, we must be sure to add less than this to avoid overcorrecting.

The largest inaccuracy is when your birthday is February 29th or March 1st. On these days, your age can be off by 0.0023, so we want to add at least this amount. With this formula, people born on leap-days celebrate their birthday on February 28th in non-leap years.

```
int(yrdif(dob, as_of_date, 'ACT/ACT') + 0.0023)
```

If you prefer leap-babies to celebrate their birthday on March 1st, then add 0.002295.

```
int(yrdif(dob, as_of_date, 'ACT/ACT') + 0.002295)
```

The `ACT/ACT` parameter tells the function to count the actual number of days between dates, accounting for years with 365 or 366 days.

This next example is a modification of a commonly used formula. If you want someone who is born on February 29th to have their non-leap year birthday be February 28th, then it could be used as an alternative.

```
int(round(as_of_date - dob)/365.25 + 0.0005, 0.001))
```

In the original formula below, the age can occasionally lag a day near a birthday following a leap year.

```
int((as_of_date - dob)/365.25)
```

There is, however, a caveat with this alternative formula. If a person’s life spans a century without a leap day (e.g., 1800, 1900, 2100), then this formula will increase their age on the day after their true birthday. This is because the formula makes an incorrect, simplifying assumption that leap years occur exactly every four years.

Adding 0.0005 and rounding to the nearest 0.001 before taking the integer part improves the results slightly, but not fully. Using the more precise number of 365.2425 days in a year yields the same, and not nearly accurate, results. Consequently, these two formulas are not accurate as often as the Kreuter one.

**FRACTIONAL AGES**

Kreuter’s formula can be altered to determine half-birthdays, or other fractional yearly ages. These two formulas yield the same results on birthdays and half-birthdays, but slightly different results on other days. When they differ, the second formula yields the lower result.

```
round((intck('MONTH', dob, as_of_date) - (day(as_of_date) < day(dob)))/12, 0.1)
round((intck('MONTH2', dob, as_of_date) - (day(as_of_date) < day(dob)))/6, 0.1)
```

These differences, however, are likely to be insignificant since they are within 0.2 of each other. The point of showing these variations is to illustrate the different ways SAS allows you to solve one problem.

The `MONTH` interval, naturally, changes whenever the name of the month changes so that there are twelve `MONTH` intervals in a calendar year. The `MONTH2` interval is two months long, and changes every other month so that there are six `MONTH2` intervals in a calendar year. The months January and February together form one `MONTH2` interval.

Attempting to alter the modification of the `YRDIF()` function (e.g., rounding to the nearest tenth) does not yield correct results. You age a few days earlier than you should.

```
round(yrdif(dob, as_of_date, 'ACT/ACT') + 0.0023, 0.1);
```

**EXAMPLE 3**

Sam was born on 7/04/1979, so his half birthday is 1/04. On 1/02/2009, using this formula, Sam is 29.5004 years old. This formula celebrates his half-birthday two days too early.

**OTHER METHODS**

If you want to know someone’s age as of December 31, the following two formulas will work. That is, what age will I turn this year? If someone is born on December 31, these formulas will both give a result of 0 on the day they were born, 1 on their first birthday, 2 on their second birthday, etc. The first example is more intuitive, as it is simple subtraction. The second formula, however, fits the model of the previous formulas, so for consistency’s sake it is a better choice.

```
year(as_of_date) - year(dob)
intck('YEAR', dob, as_of_date)
```
In general, when choosing a solution, not only should yielding the proper answer be a concern, but also consistency with the rest of the program, ease of reading, and ease of maintaining. Too often programmers fail to realize that someone else will be reading and maintaining their program. Having consistent and clear code allows others to understand what you are trying to do.

Similarly, the following formula can be used to determine your age as of the first day of a month. The age only changes in your birth month. Leap day birthdays are celebrated in February, whether or not it is a leap year.

\[
\text{int}(\text{intck('MONTH', dob, as_of_date) / 12})
\]

**AGE IN MONTHS**

Instead of knowing how many years someone has aged, you may be interested in knowing how many months they have aged.

This following formula acts like the Kreuter formula, but in terms of months not years. With this formula, someone’s age changes on the day of their birthday in each month. If someone was born on the last day of a month, then in the months that have fewer days than their birth month, their age will change the first day of the next month.

\[
\text{intck('MONTH', dob, as_of_date) - (day(as_of_date) < day(dob))}
\]

**EXAMPLE 4**

Joy was born on May 31st 2008. Her age in months as of June 30th 2009 is 12, and her age in months as of July 1st, 2009 is 13. This is expected, because in the case of people born on February 29th, the Kreuter formula celebrates their birthday on March 1st in non-leap years. This example is a similar situation in that a month does not always have the same number of days as a birth month.

By using just the \text{INTCK()} function, you can calculate someone’s monthly age as of the last day of a month. That is, assume everyone is born on the first day of their birth month.

\[
\text{intck('MONTH', dob, as_of_date)}
\]

This next formula is another modification of a commonly used formula seen above. Unfortunately, not only does it yield incorrect results when spanning centuries without leap days, it sometimes yields incorrect results near half birthdays.

\[
\text{int(round}((\text{as_of_date} - \text{dob})/(365.25/12)+0.0005, 0.01))
\]

The natural alteration to the modification of the \text{YRDIF()} function to calculate age in months (e.g., multiplying by twelve and taking the integer part) does not yield correct results. It also can be incorrect by a month near half birthdays.

\[
\text{int(round}((\text{yrdif(dob, as_of_date, 'ACT/ACT')} + 0.0023) * 12, 0.001))
\]

**EXAMPLE 5**

Billy was born on March 29th 2008, so he should turn 18 months on September 29th, 2009. However, this formula has him turn 18 months one day early.

**TEST CODE**

This code generates results via numerous cases of the presented formulas.

\[
\%macro calc_ages();
\* INTEGER AGES: ;
\text{age1} = \text{yrdif(dob, as_of_date, 'ACT/ACT');}
\text{age2} = \text{int((intck('MONTH', dob, as_of_date) - (day(as_of_date) < day(dob))) / 12);}
\text{age3} = \text{int(yrdif(dob, as_of_date, 'ACT/ACT') + 0.0023);}
\text{age4} = \text{int(yrdif(dob, as_of_date, 'ACT/ACT') + 0.002295);}
\text{age5} = \text{int(round((as_of_date - dob)/365.25 + 0.0005, 0.001));}
\text{age6} = \text{int((as_of_date - dob)/365.25);}
\* FRACTIONAL AGES: ;
\text{age7} = \text{round((intck('MONTH', dob, as_of_date) - (day(as_of_date)<day(dob)))12,0.1);}
\text{age8} = \text{round((intck('MONTH2', dob, as_of_date) - (day(as_of_date)<day(dob)))6,0.1);}
\text{age9} = \text{round(yrdif(dob, as_of_date, 'ACT/ACT') + 0.0023, 0.1));}
\%endmacro;
\]
* OTHER METHODS: 

age10 = year(as_of_date) - year(dob);
age11 = intck('YEAR', dob, as_of_date);
age12 = int(intck('MONTH', dob, as_of_date) / 12);

* AGE IN MONTHS: 

age13 = intck('MONTH', dob, as_of_date) - (day(as_of_date) < day(dob));
age14 = intck('MONTH', dob, as_of_date);
age15 = int(round((as_of_date - dob)/(365.25/12) + 0.0005, 0.01));
age16 = int(round((yrdif(dob, as_of_date, 'ACT/ACT') + 0.0023) * 12, 0.001));
%mend calc_ages;

%LET age_vars = age1-age16;

data age_test;
    /* Example 1: Sally is born on 2/28/2007. On her 1st birthday, YRDIF() says she is 0.9996 years old. However, on her 2nd & 3rd birthdays, YRDIF() is precisely correct. */
    dobj = '28FEB2007'd; as_of_date = '28FEB2008'd; %calc_ages(); output;
    dobj = '28FEB2007'd; as_of_date = '28FEB2009'd; %calc_ages(); output;
    dobj = '28FEB2007'd; as_of_date = '28FEB2010'd; %calc_ages(); output;
    call missing(dobj, as_of_date, of &age_vars.); output;

    /* Example 2: Fred is born on 5/29/2008. On his 1st birthday, YRDIF() says he is 0.99838 years old. Similarly, on his 2nd birthday, his age is given as 1.99838. Since his 4th birthday is in a leap year, as is his year of birth, YRDIF() accurately gives his age as 4. */
    dobj = '29MAY2008'd; as_of_date = '29MAY2009'd; %calc_ages(); output;
    dobj = '29MAY2008'd; as_of_date = '29MAY2010'd; %calc_ages(); output;
    dobj = '29MAY2008'd; as_of_date = '29MAY2012'd; %calc_ages(); output;
    call missing(dobj, as_of_date, of &age_vars.); output;

    /* Example 3: Sam was born on 7/04/1979, so his half birthday is 1/04. On 1/02/2009, using this formula, Sam is 29.5004 years old. This formula celebrates his half-birthday two days too early. */
    dobj = '04JUL1979'd; as_of_date = '02JAN2009'd; %calc_ages(); output;
    dobj = '04JUL1979'd; as_of_date = '04JAN2009'd; %calc_ages(); output;
    call missing(dobj, as_of_date, of &age_vars.); output;

    /* Example 4: Joy was born on May 31st 2008. Her age in months as of June 30th 2009 is 12, and her age in months as of July 1st, 2009 is 13. This is expected. */
    dobj = '31MAY2008'd; as_of_date = '30JUN2009'd; %calc_ages(); output;
    dobj = '31MAY2008'd; as_of_date = '01JUL2009'd; %calc_ages(); output;
    call missing(dobj, as_of_date, of &age_vars.); output;

    /* Example 5: Billy was born on March 29th 2008, so he should turn 18 months on September 29th, 2009. However, this formula has him turn 18 months one day early. */
dob = '29MAR2008'd;  as_of_date = '28SEP2009'd;  %calc_ages();  output;
dob = '29MAR2008'd;  as_of_date = '29SEP2009'd;  %calc_ages();  output;
call missing(dob, as_of_date, of &age_vars.);  output;

/* If born on 2/29, then when is birthday in non-leap years? */
dob = '29FEB1980'd;  as_of_date = '27FEB1983'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '28FEB1983'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '02MAR1983';  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '27FEB1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '28FEB1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '29FEB1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '01MAR1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '02MAR1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '27FEB1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '28FEB1984'd;  %calc_ages();  output;
dob = '29FEB1980'd;  as_of_date = '29FEB1984'd;  %calc_ages();  output;
call missing(dob, as_of_date, of &age_vars.);  output;

/* age1, age5, age6: Sometimes, ages a day too late. */
dob = '28FEB2007'd;  as_of_date = '27FEB2008'd;  %calc_ages();  output;
dob = '28FEB2007'd;  as_of_date = '28FEB2008'd;  %calc_ages();  output;
dob = '28FEB2007'd;  as_of_date = '29FEB2008'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '28MAY2009'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '29MAY2009'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '30MAY2009'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '28MAY2010'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '29MAY2010'd;  %calc_ages();  output;
dob = '29MAY2008'd;  as_of_date = '30MAY2010'd;  %calc_ages();  output;
call missing(dob, as_of_date, of &age_vars.);  output;

/* age5, age6: Sometimes, ages a day too late. */
/* age10, age11: Ages as of the last day of year. */
dob = '31DEC2002'd;  as_of_date = '30DEC2010'd;  %calc_ages();  output;
dob = '31DEC2002'd;  as_of_date = '31DEC2010'd;  %calc_ages();  output;
dob = '31DEC2002'd;  as_of_date = '01JAN2011'd;  %calc_ages();  output;
dob = '31DEC2008'd;  as_of_date = '30DEC2009'd;  %calc_ages();  output;
dob = '31DEC2008'd;  as_of_date = '31DEC2009'd;  %calc_ages();  output;
dob = '31DEC2008'd;  as_of_date = '01JAN2010'd;  %calc_ages();  output;
call missing(dob, as_of_date, of &age_vars.); output;

/* age5: Sometimes, ages a day too early, if birthday is in Jan/Feb. */
dob = '25FEB2008'd;   as_of_date = '23FEB2009'd;   %calc_ages();   output;
dob = '25FEB2008'd;   as_of_date = '24FEB2009'd;   %calc_ages();   output;
dob = '25FEB2008'd;   as_of_date = '25FEB2009'd;   %calc_ages();   output;
dob = '25FEB2008'd;   as_of_date = '26FEB2009'd;   %calc_ages();   output;
call missing(dob, as_of_date, of &age_vars.); output;

/* age9: Sometimes, ages on the 1st day of the month. */
dob = '12APR2007'd;   as_of_date = '30MAR2008'd;   %calc_ages();   output;
dob = '12APR2007'd;   as_of_date = '31MAR2008'd;   %calc_ages();   output;
dob = '12APR2007'd;   as_of_date = '01APR2008'd;   %calc_ages();   output;
call missing(dob, as_of_date, of &age_vars.); output;

/* age7, age8: Determine 1/2 birthdays. */
dob = '12JAN2007'd;   as_of_date = '11JUL2009'd;   %calc_ages();   output;
dob = '12JAN2007'd;   as_of_date = '12JUL2009'd;   %calc_ages();   output;
dob = '12JAN2007'd;   as_of_date = '13JUL2009'd;   %calc_ages();   output;
dob = '07MAY2000'd;   as_of_date = '06NOV2001'd;   %calc_ages();   output;
dob = '07MAY2000'd;   as_of_date = '07NOV2001'd;   %calc_ages();   output;
dob = '07MAY2000'd;   as_of_date = '08NOV2001'd;   %calc_ages();   output;
call missing(dob, as_of_date, of &age_vars.); output;

/* age12, age14: Ages on 1st day of the month. */
dob = '12FEB2007'd;   as_of_date = '27FEB2008'd;   %calc_ages();   output;
dob = '12FEB2007'd;   as_of_date = '28FEB2008'd;   %calc_ages();   output;
dob = '12FEB2007'd;   as_of_date = '01MAR2008'd;   %calc_ages();   output;
dob = '12FEB2007'd;   as_of_date = '02MAR2008'd;   %calc_ages();   output;
dob = '12FEB2007'd;   as_of_date = '01MAY2008'd;   %calc_ages();   output;
dob = '12FEB2007'd;   as_of_date = '02MAY2008'd;   %calc_ages();   output;
call missing(dob, as_of_date, of &age_vars.); output;

/* age13: Ages on birth day of the month. */
dob = '07OCT2005'd;   as_of_date = '06SEP2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '07SEP2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '08SEP2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '06OCT2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '07OCT2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '08OCT2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '06NOV2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '07NOV2006'd;   %calc_ages();   output;
dob = '07OCT2005'd;   as_of_date = '08NOV2006'd;   %calc_ages();   output;
call missing(dob, as_of_date, of &age_vars.); output;

    /* age15: Incorrect results when spanning centuries without leap days.*/
    doob = '28FEB1897'd;   as_of_date = '27FEB1901'd;   %calc_ages();   output;
    doob = '28FEB1897'd;   as_of_date = '28FEB1901'd;   %calc_ages();   output;
    doob = '28FEB1897'd;   as_of_date = '01MAR1901'd;   %calc_ages();   output;
run;

proc print data = age_test label;
    var dob as_of_date age1-age6 age10-age12;
    format dob as_of_date yymmdd10.;
    title "Examples of ways to calculate age (Integer Birthdays).";
run;

proc print data = age_test label;
    var dob as_of_date age7-age9 age13-age16;
    format dob as_of_date yymmdd10.;
    title "Examples of ways to calculate age (Fractional Birthdays & Month Birthdays).";
run;

CONCLUSION
If you are concerned with precision in calculating a yearly age, use either Kreuter’s formula or the modification to the built-in function YRDIF(). Both yield the same, accurate results.

REFERENCES

ACKNOWLEDGMENTS
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.

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

    Lloyd Butler
    Health Dialog
    2 Monument Square
    Portland, ME 04101
    lbutler@healthdialog.com