

Age Calculation

Billy Kreuter, University of Washington

This macro returns an integer representing an age (or similar interval) calculated according to common usage -- i.e., incremented exactly on a person's birthday (or other anniversary), and always correct in all leap years and non-leap years.

The birthday should be the first argument and the date of interest, on which the age will be reckoned, should be the second. The macro call should be on the right-hand side of an assignment statement. As the macro expands to a fairly compact expression, there is no error checking and the user should ensure that the first argument is always arithmetically less than the second.

In the following example, the variable AGE will be assigned 41. Either or both arguments, may, of course, be variables, expressions or a function (e.g., today()) instead of constants.

```
age = %age('24mar55'd, '15may96'd)
```

The algorithm and a slightly different form of the macro were conceived by Billy Kreuter, University of Washington, billyk@u.washington.edu. Ian Whitlock (whitloil@westatpo.westat.com) and "Doc" Muhlbaier (muhlba001@mc.duke.edu) also made contributions.

```
/* returns age in integer years on &DATE elapsed since &BIRTH */
%macro age(birth,date);
floor((intck('month',&birth,&date)-(day(&date)<day(&birth)))/12);
%mend age;
```

'When Does the Sun Rise and Fall?'

Billy Kreuter, University of Washington

This program uses proc calendar to create a display of the current month. Each day's cell shows the times of sunrise and sunset for the location which is named in the TITLE statement and whose coordinates appear in the CARDS data. The three numbers in the CARDS statement are, respectively, latitude in degrees (north is positive), longitude in degrees (east is positive), and time zone in hours (east is positive; do not correct it for daylight savings; the US eastern time zone is thus always -5).

The program computes whether daylight savings time applies on each day according to current U.S. daylight-savings law; however, locations on year-round standard time are unknown to the program and will have times displayed with daylight savings applied during the usual months.

Other months than the current one, or multiple months, can be computed by making minor modifications to the do-loop code.

The algorithm comes from a publication of the United States Naval Observatory. During all times in the current and the next century, it should be accurate to within three minutes for latitudes less extreme than 60 degrees north or south. This accuracy is with respect to publications designed for greater accuracy than this algorithm; by convention, these are the times that the edge of the sun first (or last) becomes visible, on the (usually fictitious) assumption that the observer's location is on the open ocean. The difference between this time and the actual time that an observer will see the sun rise or set is determined primarily by what objects may be present at the observer's horizon, by the observer's altitude and by atmospheric conditions (some of which are estimated by the algorithm employed by sunset.sas and by more accurate algorithms).

The code was formatted using SI's version of sas-mode.el for Emacs 19.30.

```

/* algorithm from US Naval Observatory, _1987 Almanac for Computers_, B5-7 */
options ps=59 ls=80;
data;
  input latitude longitud timezone;
  array sun(0:1) sunset sunrise;
  pi=arccos(-1);
  radian=pi/180;
  phi=latitude*radian;
  lambda=longitud/15;
  do date=intnx('month',today(),0) to intnx('month',today(),1)-1;
    n=date-intnx('year',date,0)+1;
    select (month(date));
      when (4) dst=date>=intnx('month',date,0)
        -weekday(intnx('month',date,0))+8;
      when (10) dst=date<intnx('month',date,1)
        -weekday(intnx('month',date,1))+1;
      otherwise dst=5<=month(date)<=9;
    end;
    do s=0 to 1;
      t=n+(18-12*s-lambda)/24;
      m=(.9856*t-3.289)*radian;
      l=m+(1.916*sin(m)+.02*sin(2*m)+282.634)*radian;
      delta=.39782*sin(l);
      sun{s}=3600*(mod(24*((1-2*s)
        *arccos((-0.01454-sin(delta)*sin(phi))
          /(cos(delta)*cos(phi)))
        /(2*pi)+s)+12*(atan(.91746*tan(l))/pi
          +(mod(floor(l*2/pi)+4,4)
            in (1,2)))-.06571*t-6.622
          -lambda+timezone+dst+48,24));
    end;
  output;
end;
format sunrise sunset hhmm7.;
keep date sunrise sunset;
cards;
47.612 -122.308 -8
;
title 'Seattle';
proc calendar legend;
  start date;
  var sunrise sunset;

```