Analyzing Age of Milestone Attainments from Daily Checklist Recordings Using SAS/STAT® Procedures

Warren O. Eaton[1] & Jennifer L. Bodnarchuk
Department of Psychology
University of Manitoba

Abstract

The ages at which children first reach developmental milestones like sitting, crawling, or walking display substantial child-to-child variability that may reveal the operation of important developmental processes. Survival analysis (also called event-history analysis) is well suited for such data because it is designed to assess time-situated events characterized by a qualitative change from one discrete state to another (e.g., from not walking to walking). Multiple definitions of state changes are possible and deserve study, but their implementation from daily recordings is computationally intensive. The SAS/STAT® procedure *Expand* can efficiently convert raw milestone data into a format for analysis with the SAS *Lifereg* procedure, one of several survival analysis procedures. This report comprises an annotated SAS program for the conversion of multiple daily checklist recordings into variously defined events suitable for input into an illustrative survival analysis.

[1] Correspondence concerning this manuscript should be addressed to the first author by e-mail at *warren.eaton@ad.umanitoba.ca* or by post at the Department of Psychology, University of Manitoba, Winnipeg, Canada, R3T 2N2.

SAS Program Statements

```
/* ------------------------------------------- */
/* SECTION A: READ DATA                        */
/* ------------------------------------------- */

/* Each person has multiple records--one for each day with multiple
milestones (see Data SAMPLE).*/
Data SAMPLE (Label='Daily checklist data, one obs per person per day');
      input ID chk_date m1 m2 m3;
      datalines;
A 12OCT2006 0 0 0
A 13OCT2006 0 . 0
A 14OCT2006 1 0 0
B 21DEC2006 1 0 0
B 22DEC2006 1 0 1
Etc.... ; run;

/* DEMOGS reads birth date and other predictors needed for later steps */
Data DEMOGS (Label='Information about individual, one obs per person') ;
      input ID birth_date Predictor1 Predictor2 Predictor3;
      datalines;
A 18SEP2006 3400 2 2.48
B 11OCT2006 2985 1 2.34
C 01JUL2006 3315 1 1.99
Etc.... ; run;

/* Next step calculates dates of first and last recording for each person */
Data CHKLSTSTAT;
      set SAMPLE; by ID; retain form_start form_end;
      if first.ID then do; *reset all retained variables for new ID;
            form_start=chk_date; form_end=.;
      end;
      if last.ID then do;
      form_end = chk_date; output;
      end;
      label form_end = 'Date of last recording on form'
        form_start = 'Date of first recording on form'; run;

/* ------------------------------------------------ */
/* SECTION B: FILL IN MISSING DAYS FOR PROC EXPAND */
/* ------------------------------------------------ */

/* PROC EXPAND ignores non-continuity in observations, so it is necessary to
      insert observations for all missing days. */
```

```
Data CLEANED (Label='Lines inserted for missing checklist days');
      set CHKLSTSTAT;   by ID;
  /* Array m has all raw data for milestone items */
      array m (*) m1 m2 m3;
  /* Create new variable (prev_date) for day before current day */
      prev_date = lag(chk_date);
  /* Create new variable for check that the previous observation
    was from the same participant. */
    prev_ID = lag(ID);
  /* Branching depending on whether previous obs was the same ID
      or whether it was the previous day or not */
     if (ID = prev_ID) and (prev_date ne (chk_date - 1))
     then do;
            /* Insert lines for missing days*/
       count = chk_date - prev_date;
       do _i_ = 1 to count;ID = ID; chk_date = prev_date + _i_;
        do _m_ = 1 to dim(m); m(_m_) = .;
        end; output cleaned;
       end;
     end;
   /* If no gaps in recording, output as is. */
    else output cleaned; run;

/* ---------------------------------------- */
/* SECTION C: USE PROC EXPAND TO PREPARE DATA */
/* ---------------------------------------- */

/* It is necessary to rule out the possibility that milestone was reached
  before data collection began. To do so, check week prior to first recorded
  instance of the milestone to see if it was not observed on at least 4 days
  in the previous week. If the milestone was observed within a week of
  starting checklist completion, then the start of the checklist is used as
  the left censoring variable (to be used in PROC LIFEREG below). */

/* Cannot sum zeros, so create new variable that changes all ones to zeros
  and all zeros to ones, missing left as missing. This will allow "zeros" to
  be counted. */

data CONVERT; set cleaned;

/* Array m has all raw data for milestone items */
      array m (*) m1 m2 m3;
/* Array reversed for sums of not attained (0) codes */
  array reversed (*) m1_0 m2_0 m3_0;
/* Reversed values calculated from milestone raw data */
```

```
   do _i_ = 1 to dim(m); *change 0's to 1's, leave missing as missing;
      reversed(_i_) = abs ( m(_i_) -1);
   end; drop _i_; run;

Proc Expand data=CLEANED out=EXPAND1
      method=none; *suppresses interpolation; by ID;
      id chk_date; *identifies the SAS date variable;

  /* Variable names with _0sum suffix contain sum of 0's over 8-day window
    (includes current day) */
      convert m1_0=m1_0sum m2_0=m2_0sum m3_0=m3_0sum
   /transform = (movsum 8); *moving sum over current & previous 7 days; run;

Data DOF (label='Dates of first attainment, 1 obs per ID'); set EXPAND1;
      by ID;
       * By ID specification needed for first.ID and last.ID vars;

  /* Array m has all raw data for milestone items */
      array m (*) m1 m2 m3;
  /* Array zeroes for sums of not attained (see previous PROC EXPAND) */
    array zeroes (*) m1_0sum m2_0sum m3_0sum;
  /* Array for date of first (DOF) attainment, variables to be output */
      array DOF (*) m1_DOF m2_DOF m3_DOF;
  /* Array for previous attainment, variables to be output */
      array PREVPASS (*) m1_PREV m2_PREV m3_PREV;
  /* Program steps below determine date of first attainment */
      retain m1_DOF m2_DOF m3_DOF m1_PREV m2_PREV m3_PREV;
  /* At beginning of new ID */
      if first.ID then do; *reset all retained variables for new ID;
            do _i_ = 1 to dim(DOF);
       DOF(_i_)=.; /* sets DOF variables to missing */
             PREVPASS(_i_)=.; /* sets previous pass to missing */
      end;
   end;
  /* For each observation */
      do _i_ = 1 to dim(DOF);
        if chk_date - 3 <= form_start /* in 1st 4 days of obs */
            and m(_i_) = 1   /* milestone observed */
        then PREVPASS(_i_)= 1; /* DOF passed previously */
        if m(_i_) = 1    /* Is current obs a 1? */
            and ZEROES(_i_) > 3 /* 4+ zeroes in previous week */
            and PREVPASS(_i_) ne 1 /* no previous pass */
             and DOF (_i_) = .  /* DOF not previously assigned */
        then DOF(_i_) = chk_date; /* DOF during checklist */
       end;
 /* At the end of ID */
```

```
      if last.ID then output;
 /* Keep only variables needed for later steps and assign labels. */
      keep ID m1_DOF -- m3_PREV;
      label m1_DOF = 'Date of 1st attain, first milestone'
       m2_DOF = 'Date of 1st attain, second milestone'
       /* etc. */;      run;

/* ------------------------------------------------- */
/* SECTION D: CREATE "AGE-OF-ATTAINMENT" (AOA) VALUES */
/* ------------------------------------------------- */

Data AOA (Label='Ages of attainment, one obs per indiv');
      merge DOF(in=DOA_OK) DEMOGS;  by ID; if DOA_OK;
      array alldates (*) m1_DOF m2_DOF m3_DOF form_start form_end;
      array allages (*) m1_AOF m2_AOF m3_AOF age_start age_end;
 /* Convert all dates to ages in weeks using birth date */
      do _i_=1 to dim(alldates);
            allages(_i_) = (alldates(_i_) - birth_date) / 7;
       end;
/* Keep and label variables needed for later steps */
      keep ID m1_AOF-- age_end;
      label m1_AOF = 'CA wks first attain, first milestone'
            /* etc. */; run;

/* ---------------------------------------------- */
/* SECTION E: CREATE CENSORING VARIABLES     */
/* ---------------------------------------------- */

Data CENSOR (Label='Censor vars by milestone for EHA analyses');
      merge DOF AOA;    by ID;
 /* Array AOF for first attainment dates */
      array AOF (*) m1_AOF m2_AOF m3_AOF;
 /* Array for variable, _PREV, indicating previous attainment, 1 */
      array PREVPASS (*) m1_PREV m2_PREV m3_PREV;
 /* Array for EHA lower boundary, an age */
      array LWR (*) m1_LWR m2_LWR m3_LWR;
 /* Array for EHA upper boundary, an age */
      array UPR (*) m1_UPR m2_UPR m3_UPR;
 /* Following section assigns lower and upper EHA boundaries
      - For uncensored data. Lower=AOA and Upper=AOA
      - For right censored data. Lower=age at completion and Upper=.
      - For left censored data, Lower=. and Upper=age at start */
      do _i_ = 1 to dim(AOF);
            /*uncensored*/
              if AOF(_i_) ne . then do;
                  LWR(_i_) = AOF(_i_); UPR(_i_) = AOF(_i_);
```

```
          end;
           /*right censored*/
             if AOF(_i_) = . and PREVPASS(_i_) = . then do;
                  LWR(_i_) = age_end; UPR(_i_) = .;
              end;
           /*left censored, PREVPASS determined earlier */
             if AOF(_i_) = . and PREVPASS(_i_) = 1 then do;
                  LWR(_i_) = .; UPR(_i_) = age_start;
             end;
     end;

/* Keep only variables needed for later steps and assign labels. */
 keep ID m1_AOF--m3_AOF m1_LWR--m3_LWR m1_UPR--m3_UPR age_start age_end;
 label m1_LWR = 'Age at lower bound, first milestone'
  m2_LWR = 'Age at lower bound, second milestone' /* etc. */
  ; run;

/* ------------------------------------------------ */
/* SECTION F: SURVIVAL ANALYSIS WITH MILESTONE M1  */
/* ------------------------------------------------ */

/* Merge censored values with predictor variables by ID */
Data EVA; merge Demogs Censor; by ID;

/* M1 is predicted by three predictors in the following example */

Proc lifereg data=EVA outest=m1outest;
  model (m1_LWR,m1_UPR) = Predictor1 Predictor2 Predictor3/ Dist=gamma;
  output censored=m1_censor p=predm1 out=m1out xbeta=m1xbeta;
  run;

  /* Run Allison lifehaz macro to obtain hazard plots
      for lifereg; available at http://www.ssc.upenn.edu/~allison/#Macros  */
  %lifehaz(outest=name1,out=name2,xbeta=name3,obsno=0); run;

  /* To export results to Excel for easier plotting, etc. *:
  proc export dbms=excel2000 replace
      outfile="C:\Survival hazard.xls";
      sheet=lifehaz_out ;  run;
```

Reference

Allison, P. D. (2010). *Survival analysis using SAS®: A practical guide(2nd ed.)*. Cary, NC: SAS Press.