Paper 89-28

Identifying Continuity in Longitudinal Data

Merle E. Hamburger, Ph.D., Northrop Grumman, Atlanta, GA &
Thomas Sukalac, CDC, Atlanta, GA

ABSTRACT

Analyzing longitudinal data (performing analyses across multiple observations) may be one of the most difficult processes facing a SAS analyst. One particularly difficult longitudinal analysis involves the identification of the extent to which an individual continuously experienced a particular event over the course of the longitudinal data. The current paper describes a public health analysis in which the RETAIN statement, as well as the FIRST. and LAST. logical variables are used to determine the length of time an HIV-infected individual has a continuously undetectable level of the virus.

INTRODUCTION

Analyzing longitudinal data is one of the most challenging tasks an analyst may be asked to do in that it is often quite challenging to conduct operations across multiple observations of the same individual. For example, you may have a dataset that contains data from a variable number of visits for each individual, with the data from each visit listed as a separate observation in the dataset. In such a dataset, it may be of interest the length of time individuals in the data set continuously meet some criteria (e.g., in a financial dataset, this may involve identifying how long individuals have continuously paid their monthly bills on time). The RETAIN, FIRST., and LAST. functions in the DATA step, however, allow one to assess this continuity relatively easily.

AN EXAMPLE

Longitudinal data are consistently used in research when investigating various public health phenomena. Thus, these datasets contain some number of observations per individual over time. To add a little spice, it would not be unusual for different individuals to have varying numbers of observations over the same time period. Of key importance to research on HIV treatments is the identification of criteria regarding the ‘success’ or ‘failure’ of a particular treatment. Often public health scientists rely on measures of disease progression (e.g., CD4 lymphocyte count or Viral Load) as an indicator of treatment success (e.g., increased CD4 count or decreased or undetectable viral load). For the purposes of this paper, I am going to discuss longitudinal data from a cohort study of HIV-infected persons for which we were interested in determining which participants had long term ‘successful’ HIV treatment using viral load as the indicator. In particular, we wanted to identify those persons in the cohort that had continuously undetectable viral loads for at least two years. The variable VBDATE is the date of viral load assessment and VLOAD is the variable indicating the number of copies of the virus in the blood system. Please note that if the virus is undetectable, then VLOAD is given a value of 1.

APPLICATION AND PROGRAMMING

We start with a longitudinal dataset VLDATA which contains the following variables: HRNID (study ID variable), VLOAD (HIV viral load), and VBDATE (date of viral load assessment). Note that each individual has a variable number (ranging from 1 to 47) of viral load observations. Referring to the SAS code below, the first step in this process is to sort the data in VLDATA by the ID variable (as it is very important for manipulating the longitudinal data in the DATA step).

```
proc sort data=VLDATA ;
        by hrnid ;
run ;
```

Now that the data are sorted, I will discuss the DATA step necessary to compute the interval of time an individual has an undetectable viral load (VLOAD = 1). The first part of the SAS code creates a new dataset, INTERVAL, using data from VLDATA. INTERVAL will contain the variables HRNID, BEGDATE, ENDDATE, INTERVAL, and UNDETECT (all variables that are computed (and explained) later in the DATA step. The BY command causes SAS to create two variables: the FIRST. and LAST. logical variables. When SAS encounters the first observation within the by group (in the code below, this would be the first observation of a particular HRNID), FIRST.HRNID is true; otherwise it is false. Similarly, when SAS encounters the last observation within the by group (in this example, the last observation for a particular HRNID), LAST.HRNID is true; otherwise it is false. It is important to know that SAS runs the entire DATA step for each observation; at the beginning of each new observation, SAS sets the values for variables to missing and then populates them according to the commands in the DATA step. The RETAIN statement is important when dealing with longitudinal data because it causes SAS to keep the values of the variables listed from one observation to the next.

```
data INTERVAL ;
        set VLDATA ;
        by hrnid ;
        retain inint begdate enddate ;
        keep hrnid begdate enddate interval undetect ;
```

The next part of the SAS code uses the FIRST. logical variable. In this case, FIRST.HRNID is used in a conditional statement that sets the value of the variable ININT to 0 if the current observation is the first occurrence of that particular HRNID (if this was NOT the first occurrence, FIRST.HRNID would be false and the conditional is not be processed).

```
        if first.hrnid then inint = 0 ;
```

Next, SAS is told to set the variable BEGDATE equal to the data of viral load assessment (VBDATE) and to set the variable ININT = 1 if ININT = 0 and the viral load was undetectable (VLOAD = 1). Thus, BEGDATE is the date at which a participant has the first instance of having an undetectable viral load and ININT becomes a flag for having an undetectable viral load (recall that the values for these variables are retained for the next observation). Note that with the IF…THEN DO command structure, it is necessary to place an END statement after the last command to be executed as a result of the conditional being true.

```
        if (inint = 0) and (vload = 1) then do ;
            begdate = vbdate ;
            inint = 1 ;
        end ;
```

This next code sets the variable ENDDATE to the date of viral load assessment (VBDATE).

```
        if inint = 1 then enddate = vbdate ;
```

The next section of code sets up another conditional statement. Specifically, if the 'undetectable' flag is turned on (ININT = 1) and the current viral load is detectable (i.e., VLOAD is > 1) then ININT is reset to 0 and the variable INTERVAL is computed as the number of years between the date variables ENDDATE and BEGDATE. (Note: ENDDATE-BEGDATE would give you the number of days between these variables. Dividing this by 365.25 gives you the number of years.) Finally, if the interval between these dates is 2 or more years (INTERVAL ge 2), then the variable UNDETECT is initialized to equal 1 and the observation is outputted. Note that both IF…then statements need END statements to close the loop.

```
    if inint = 1 and vload > 1 then do ;
        inint = 0;
        interval = (enddate - begdate) / 365.25 ;
        if interval ge 2 then do ;
            undetect = 1 ;
            output;
        end ;
    end ;
```

The following code uses the logical variable LAST.HRNID. If the current observation being analyzed by SAS is the last observation for a particular HRNID and ININT = 1, then the code computes the interval between ENDDATE and BEGDATE. If the value of INTERVAL is 2 or more years, then UNDETECT is initialized to 1 and the data is outputted.

```
    if last.hrnid and inint = 1 then do ;
        interval = (enddate - begdate) / 365.25 ;
        if interval ge 2 then do ;
            undetect = 1 ;
            output;
        end ;
    end ;
```

This final code simply formats the date and interval variables.

```
    format begdate mmddyy10.  enddate mmddyy10.
        interval 4.2 ;

run ;
```

Using this procedure, your final dataset will contain only those individuals, from the original dataset, who had undetectable viral loads for at least continuous 2 years. Here is the full code (this is the same as above, but without the commentary separating the sections.

```
data INTERVAL ;
        set VLDATA ;
        by hrnid ;
        retain inint begdate enddate ;
        keep hrnid begdate enddate interval undetect ;

if first.hrnid then inint = 0 ;

        if (inint = 0) and (vload = 1) then do ;
                begdate = vbdate ;
                inint = 1 ;
        end ;

if inint = 1 then enddate = vbdate ;

        if inint = 1 and vload > 1 then do ;
                inint = 0;
                interval = (enddate - begdate) / 365.25 ;
                if interval ge 2 then do ;
                        undetect = 1 ;
                        output;
                end ;
        end ;

        if last.hrnid and inint = 1 then do ;
                interval = (enddate - begdate) / 365.25 ;
                if interval ge 2 then do ;
                        undetect = 1 ;
                        output;
                end ;
        end ;

        format begdate mmddyy10. enddate mmddyy10.
                interval 4.2 ;

run ;
```

CONCLUSION

This code, though used here within a public health example, has applications across disciplines that deal with longitudinal data. Whether it is investigating individuals who consistently pay their bills on time or determining who consistently shows up to work late, this code should be helpful if you are collecting longitudinal data, and are interested in identifying the extent to which you are getting consistent data across time.

CONTACT INFORMATION

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

Merle E. Hamburger
CDC-P
1600 Clifton Rd., MS E-45
Atlanta, GA 30333
(404) 639-5263
(404) 639-6127 (fax)
mhamburger@cdc.gov