

Duration Calculation from a Clinical Programmer's Perspective

Alice M. Cheng, Scios Inc., Fremont, CA

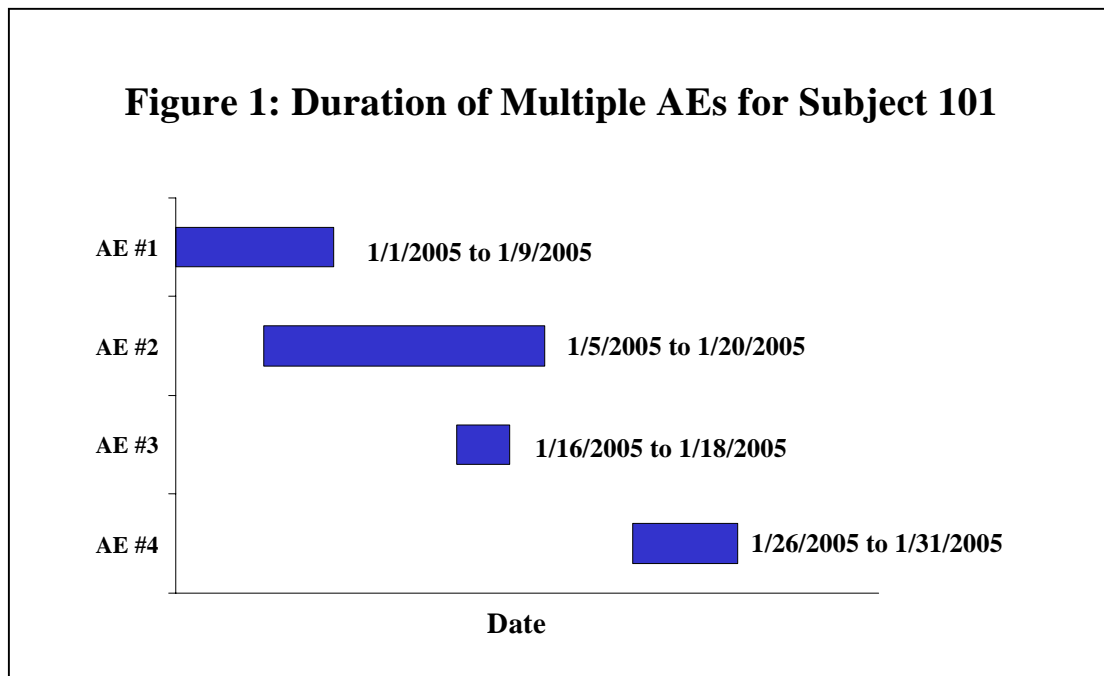
ABSTRACT

Calculation of duration is common in clinical reporting. Often in clinical trials, the start dates and stop dates of adverse events or medication usage are recorded. Clinical programmers are required to compute the number of distinct days in these time intervals for each subject. When a subject has multiple time intervals, gaps and overlaps in these time intervals can be a major challenge in duration calculation. In this paper, the author explores different methods to compute duration. A brief discussion of the pros and cons of these methods is provided.

Keywords: Duration, Time Intervals, Gaps, Overlaps

INTRODUCTION

In clinical trials, a subject may have experienced multiple adverse events (AEs) with different degrees of severity or may have taken different medication at various time intervals. Consider the situation depicted in Figure 1 below.



Subject 101 has experienced adverse events in four different time intervals. Some of these time intervals overlap each other. One is entirely embedded within another interval. Another, the last interval, is disjoint from other intervals. Because of overlaps and gaps among intervals, one cannot compute the number of distinct days by simply adding the total number of days in each interval. In this example, the actual number of distinct days with adverse events is 26; however, by adding number of days in each interval, the total number of days is 34! In other words, the duration has been inflated by 8 days!

In this paper, the author introduces 4 methods to count the number of distinct days among time intervals. Some methods are easier to comprehend while others may reduce the amount of execution time and use less memory. Although in this example, duration calculation is in days, the same concept can be applied when minutes or even seconds are used. All intervals are assumed to have definite start and stop dates. Open-ended intervals are not considered here. Of course, there are numerous methods to calculate duration; the 4 methods introduced here are not meant to be exhaustive. Instead, they are methods for you to consider. At the end of this paper, the author will briefly discuss the advantages and disadvantages of the methods introduced.

DATA

Without loss of generality, only critical data from one subject are considered in this example. For simplicity, visit numbers are not included in these data, though in clinical trials, programmers are often required to compute distinct number of AE days for each subject-visit. Based on Figure 1, the AE_DUR dataset has been created.

AE_DUR Data Set

<u>OBS</u>	<u>SUBJID</u>	<u>STARTDATE</u>	<u>STOPDATE</u>
1	101	01JAN2005	09JAN2005
2	101	05JAN2005	20JAN2005
3	101	16JAN2005	18JAN2005
4	101	26JAN2005	31JAN2005

Note that SUBJID is of 8. format. STARTDATE and STOPDATE are both of date9. format. AE_DUR dataset is said to have a vertical representation of the data since repeated STARTDATE and STOPDATE are stored in another observation instead of repeated fields.

For some methods, it will be easier to have a dataset like AE_DURH below --- a horizontal representation of the AE_DUR data set. Using repeated fields for start dates and stop dates.

AE_DURH Data Set

<u>OBS</u>	<u>SUBJID</u>	<u>START1</u>	<u>START2</u>	<u>START3</u>	<u>START4</u>
1	101	01JAN2005	05JAN2005	16JAN2005	26JAN2005
<u>OBS</u>		<u>STOP1</u>	<u>STOP2</u>	<u>STOP3</u>	<u>STOP4</u>
1		09JAN2005	20JAN2005	18JAN2005	31JAN2005

Such horizontal presentation can be easily achieved by the following SAS[®] code:

```
*--- Transpose data from Vertical Representation to Horizontal Representation. ---*;
proc sort data=AE_DUR;
  by SUBJID STARTDATE;
run;

proc transpose data=AE_DUR out=STARTDT (drop=_name_) prefix=START;
  by SUBJID;
  var STARTDATE;
run;

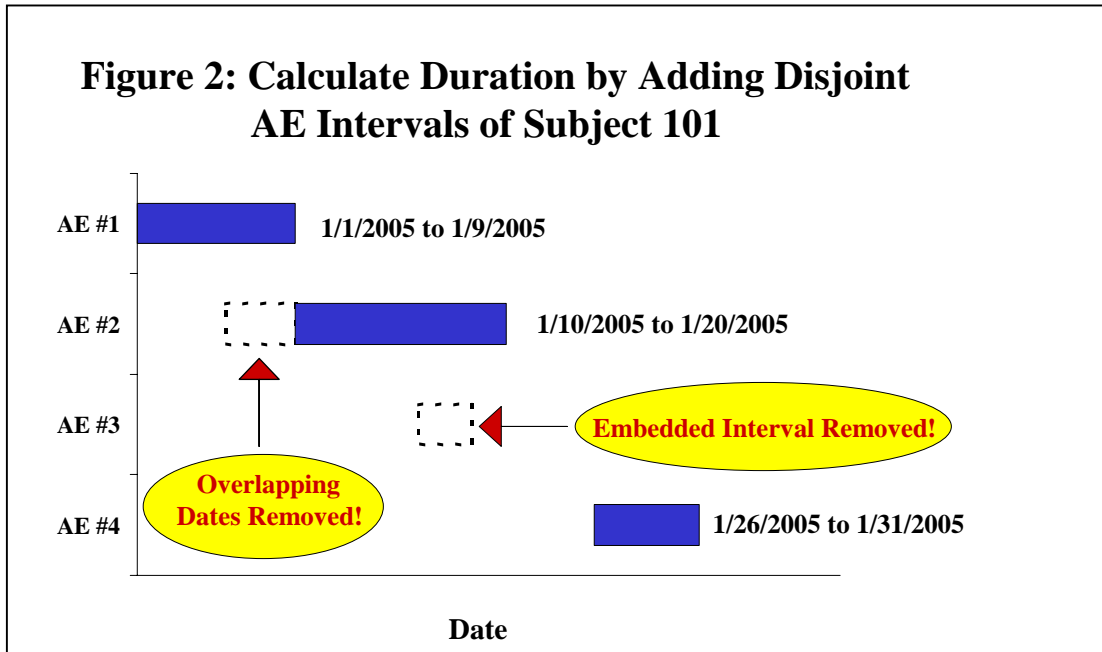
proc transpose data=AE_DUR out=STOPDT (drop=_name_) prefix=STOP;
  by SUBJID;
  var STOPDATE;
run;

data AE_DURH;
  merge STARTDT STOPDT;
  by SUBJID;
run;
```

For Methods 1 and 2 discussed below, it is critical that observations from each subject are sorted in ascending order of the STARTDATE in AE_DUR dataset so that START1 <= START2 <=...<=STARTn in AE_DURH dataset.

METHOD 1: ADD DISJOINT INTERVALS

Strategy: Convert overlapping intervals to disjoint intervals. Remove embedded intervals. Add up days from disjoint intervals to get the total duration.



As illustrated in Figure 2, overlapping dates from 1/5/2005 to 1/9/2005 are removed from the second AE interval. Since the third AE interval is embedded within the second AE interval, this third interval is entirely removed. Now the total duration can be easily calculated by adding the days from the disjoint intervals. In this example,

$$\begin{aligned} \text{Total Number of AE Days} &= 9 \text{ days (from 1/1/2005 to 1/9/2005)} + 11 \text{ days (from 1/10/2005 to 1/20/2005)} + \\ &\quad 6 \text{ days (from 1/26/2005 to 1/31/2005)} \\ &= 26 \text{ days} \end{aligned}$$

SAS CODE FOR METHOD 1

```
*--- Method 1: Add Disjoint Intervals. ---*;
data METHOD1;
retain TOTDUR;
set AE_DURH; *--- Use Horizontal Data Representation. ---*;
array START{*} START:;
array STOP{*} STOP:;

TOTDUR=STOP1-START1+1;

do i=2 to dim(START);

do j=1 to i-1;

*--- Make overlapping but not embedded intervals disjoint. ---*;
if . < START(i) <= STOP(j) and STOP(i) > STOP(j) > . then
do;
START(i)=STOP(j)+1;
end;

*--- Embedded in previous interval. Will not contribute to duration. ---*;
else
do;
```

```

        if . < START(i) <= STOP(j) and . < STOP(i) < STOP(j) then
        do;
            START(i)=.;
            STOP(i)=.;
        end;
    end;
end;

if START(i) ne . and STOP(i) ne . then DUR=STOP(i)-START(i)+1;
else DUR=0;
TOTDUR+DUR;
end;
label TOTDUR='DURATION |(DAYS)';
run;

proc print data=METHOD1 label split='|';
title 'Result from Method 1';
var SUBJID TOTDUR;
run;

```

SAS OUTPUT FROM METHOD 1

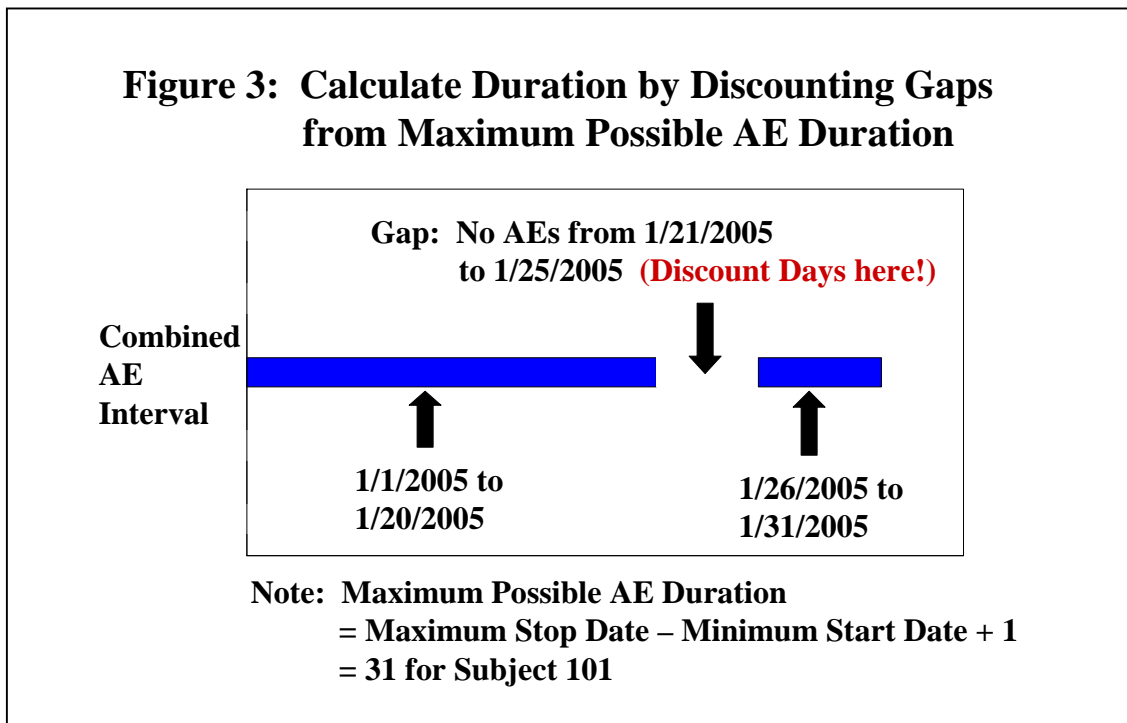
Result from Method 1

OBS	SUBJID	DURATION (DAYS)
1	101	26

Caevet: For Method 1 to work, it is important that the STARTDATE in AE_DUR dataset is in ascending order (or START1 <= START2 <= ... <= STARTn in AE_DURH dataset); otherwise, the number of distinct days in the disjoint intervals would not be counted correctly!

METHOD 2: DISCOUNT DAYS IN THE GAPS

Strategy: Find the minimum start date and the maximum stop date. Get the maximum possible duration based on these 2 dates. Subtract the gaps (days that are not in any of the time intervals) from the maximum possible duration.



Based on Figure 3, for Subject 101, the minimum start date is 1/1/2005 and the maximum stop date is 1/31/2005. Maximum Possible Duration is defined as:

Maximum Possible Duration = Maximum Stop Date – Minimum Start Date +1

Hence, for Subject 101, the maximum possible AE duration = 1/31/2005-1/1/2005+1=31. However, we do have to get rid of the gaps so that

Actual Duration = Maximum Possible Duration – Gaps

As a result, for Subject 101, we will exclude the 5 days (1/21/2005 to 1/25/2006) in which the subject has no AEs. The actual duration is 31-5=26 days.

SAS CODE FOR METHOD 2

```
*--- Method 2: Discount Days in the Gaps. ---*;
data METHOD2;
  set AE_DURH;    *--- Use Horizontal Data Representation. ---*;
  array START{*} START:;
  array STOP{*} STOP:;

  *--- Find the Maximum Possible Duration ignoring the gaps. ---*;
  DURATION= max (of STOP:) - min (of START:)+1;
  do i=1 to dim(START)-1;

    *--- Find the Maximum Stop Date from first to the i-th intervals. ---*;
    do j=1 to i;
      if j=1 then MAX_PREV_STOP=STOP(1);
      if j>1 and STOP(j) > MAX_PREV_STOP then MAX_PREV_STOP=STOP(j);
    end;

    *--- Discount days due to the gaps with no AEs. ---*;
    if START{i+1}-MAX_PREV_STOP > 0 then
      DURATION=DURATION-(START{i+1}-MAX_PREV_STOP)+1;
  end;
  label DURATION='DURATION|(DAYS)';
run;

proc print data=METHOD2 label split='|';
  var SUBJID DURATION;
  title 'Result from Method 2';
run;
```

SAS OUTPUT FROM METHOD 2

```
Result from Method 2

      OBS      SUBJID      DURATION
              (DAYS)
      1          101          26
```

Caevet: For Method 2 to work, it is important that the STARTDATE in AE_DUR dataset is in ascending order (or START1 <= START2 <= ... <= STARTn in AE_DURH dataset); otherwise, the number of days in the gap would not be counted correctly!

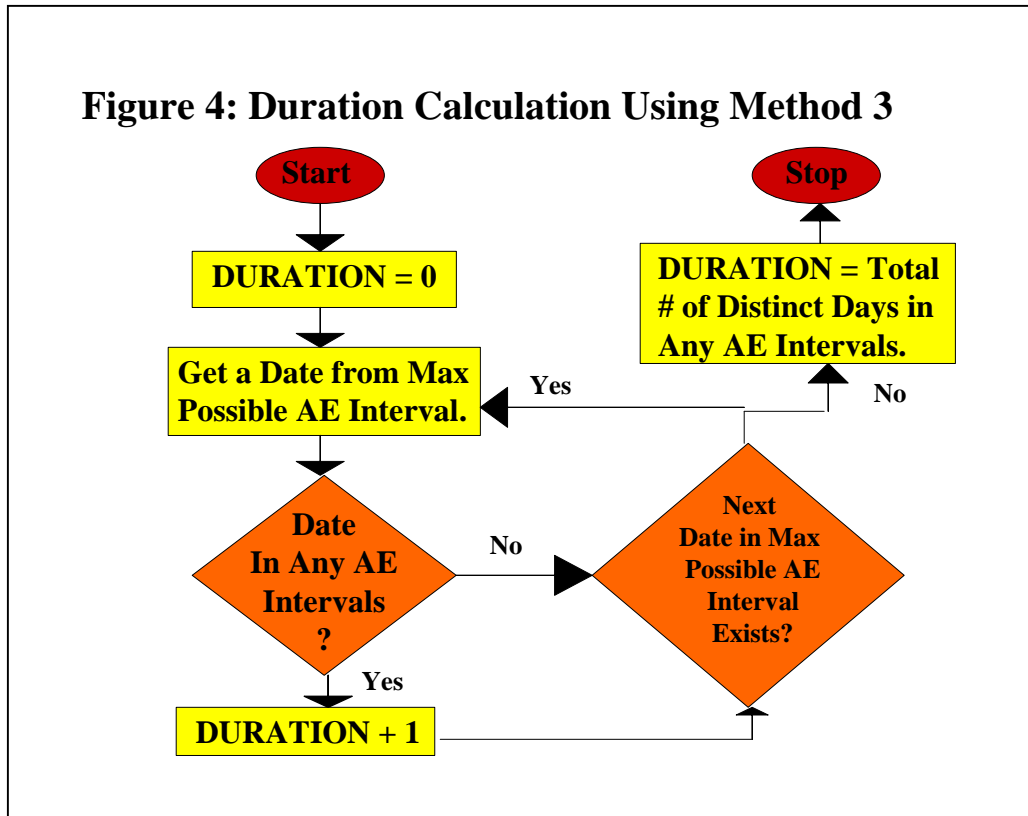
METHOD 3: CHECK ONE DAY AT A TIME

Strategy: Find the maximum possible AE interval. Check for each date in the maximum possible interval if a date is within any of the time intervals. Accumulate the total number of distinct dates within any interval to get the duration.

The maximum possible AE Interval begins with the minimum start date and ends with the maximum stop date. In other words.

Maximum Possible AE Interval = [Minimum Start Date, Maximum Stop Date]

In our example, for Subject 101, the minimum start date is 1/1/2005 and the maximum stop date is 1/31/2005. Hence, the maximum possible AE interval is [1/1/2005, 1/31/2005]. As illustrated in Figure 4, Method 3 checks each date within the maximum possible AE Interval to see if the date concerned exists in any of the 4 intervals (i.e., [1/1/2005, 1/9/2005], [1/5/2005, 1/20/2005], [1/16/2005, 1/18/2005] and [1/26/2005, 1/31/2005]). Accumulate 1 day to the total duration if a date can be found in any of the 4 AE intervals.



SAS CODE FOR METHOD 3

```

*--- Method 3: Check One Day At A Time. ---*;
data METHOD3;
  retain i DURATION;
  set AE_DURH; *--- Use Horizontal Data Representation. ---*;
  DURATION=0;

  array START{*} START:;
  array STOP{*} STOP:;

  *--- Find the Maximum Possible AE Interval. ---*;
  INTV_NUM=dim(START);
  MINSTART=min(of START:);
  MAXSTOP= max(of STOP:);

  *--- Check one day at a time from the Maximum Possible AE Interval. ---*;
  do XDATE=MINSTART to MAXSTOP;
    i=1;
    IN_INTERVAL=0;

    *--- Check if a date is within any interval. ---*;
    *--- Stop checking once a date has been found within an interval or ---*;
    *--- all dates have been checked and the date has not yet been found. ---*;
    do while (IN_INTERVAL=0 and i <= INTV_NUM);
      if (START(i) <= XDATE <= STOP(i)) then IN_INTERVAL=1;
      i+1;
    end;
    if IN_INTERVAL=1 then DURATION+1;
  end;
end;

```

```

label MINSTART='Minimum Start Date'
      MAXSTART='Maximum Stop Date'
      DURATION='Duration|(Days)';
run;

proc print data=METHOD3 label split='|';
var SUBJID DURATION;
title 'Result from Method 3';
run;

```

SAS OUTPUT FROM METHOD 3

Result from Method 3

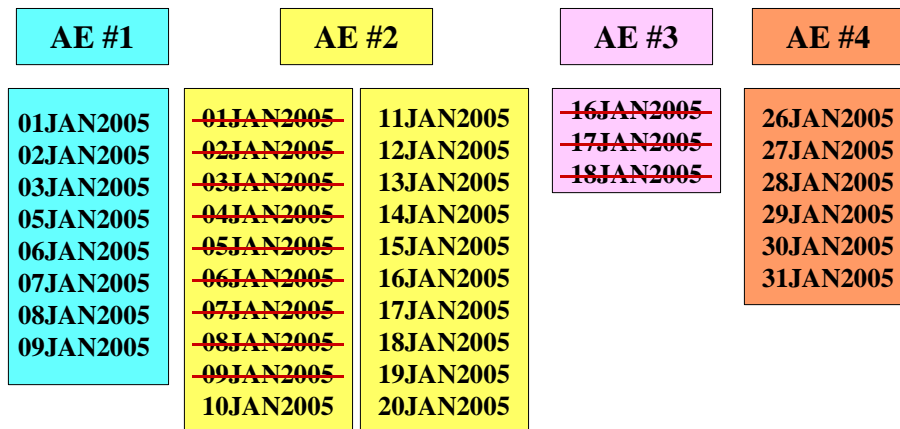
OBS	SUBJID	DURATION (DAYS)
1	101	26

Note: Unlike Methods 1 and 2, Method 3 does not require start date to be sorted in ascending order. That is, START1 <= START2<= ... <=STARTn in AE_DURH dataset is not required. Of course, one has to make sure START1 and STOP1 represent the first AE interval, START2 and STOP2 represent the second AE interval, etc.

METHOD 4: SIMPLY COUNTING

Strategy: Count every distinct date within any time intervals only once.

Figure 5: Simply Count Distinct Date from All Intervals Once



Note: Repeated dates are strike-out and are not counted.

Internally, each SAS date is represented by a number. January 1, 1960 is represented by the number zero. The internal value for each SAS date represents the number of days from January 1, 1960. Any date prior to January 1, 1960 has a negative value and any date after January 1, 1960 has a positive value. For instance,

- 2 stands for 30DEC1959
- 1 stands for 31DEC1959
- 0 stands for 01JAN1960
- 1 stands for 02JAN1960
- 2 stands for 03JAN1960

Because of this reason, the following simple code generates an observation for every date from start date to stop date of each interval.

```
data AE_DURR;
  set AE_DUR; *--- Use Vertical Data Representation. ---*;
  do XDATE=STARTDATE to STOPDATE;
    output;
  end;
run;
```

Now use **count (distinct XDATE)** and **group by SUBJID** in PROC SQL to produce the number of distinct AE days for the subject concerned.

SAS CODE FOR METHOD 4

```
*--- Method 4: Simply Counting. ---*;
*--- Generate an observation for each date within any interval. ---*;
data AE_DURR;
  set AE_DUR;
  do XDATE=STARTDATE to STOPDATE;
    output;
  end;
run;

*--- Count only the distinct dates within any intervals. ---*;
proc sql noprint;
  create table METHOD4 as
  select SUBJID, count (distinct XDATE) as DURATION label='Duration|(Days)'
  from AE_DURR
  group by SUBJID
  order by SUBJID
  ;
quit;

proc print data=METHOD4 label split='|';
  var SUBJID DURATION;
  title 'Result from Method 4';
run;
```

SAS OUTPUT FROM METHOD 4

Result from Method 4		
OBS	SUBJID	DURATION (DAYS)
1	101	26

Note: Like Method 3, Method 4 does not require start date to be sorted in ascending order. Unlike previous 3 methods, Method 4 is slightly more convenient to use in a vertical data representation (AE_DUR dataset). It is also the only method that requires the generation of an additional temporary dataset.

COMPARISON

So, which is the best method? There is no easy answer to this question. It all depends on the circumstances and your priority! Just by examining the code, Method 4 (*Simply Counting*) seems to be the simplest and easiest to comprehend. It does not have the nested DO LOOPS as the other 3 methods and it does not require data to be sorted in ascending order of start date as in Methods 1 and 2. However, if the duration has been lengthy, there is a good chance that Method 4 results in a huge temporary dataset in Data Step AE_DURR, which takes up considerable amount of memory. Therefore, under such circumstances, you should consider using the other 3 methods which do not require an additional temporary dataset, or at least, to save memory, bring in only the necessary variables in AE_DURR data step when using Method 4! Of course, the amount of computation will slightly increase for Method 4, should the data come in horizontal representation!

Method 3 (*Check one day at a time*) is the second easiest to understand. Like Method 4, it also has the advantage that the input data does not have to be sorted in an ascending order of start date. However, it may require a lot of iterations because every date within the maximum possible AE duration has to be checked against the individual intervals. Checking only ends when the date has been identified in an individual interval or when SAS completes

checking all individual AE intervals for the subject without finding the date concerned in any intervals! The number of iterations can be considerable if the maximum possible AE interval is lengthy and/or there are a lot of individual intervals!

Method 1 (*Add Disjoint Intervals*) and Method 2 (*Discount Days in the Gap*) are not as comprehensible as Methods 3 and 4. However, provided that the incoming data are in horizontal representation, that there are not too many individual intervals and data have already been sorted in ascending order of start date, even with nested DO LOOPS, both methods may actually result in less iterations, shorter execution time and less memory usage!

All in all, there is no one single method that is the best under all circumstances. The nature, structure and ordering of the data, the degree of comprehensibility, as well as, your goals, resources and priority should serve as guidelines on determining the best method to use under the circumstances.

CONCLUSION

This paper explores 4 different approaches to duration calculation. Although in this example, duration is in days, the same concept can be applied when minutes and seconds are used in measuring duration. The methods explored in this paper are, by no means, exhaustive. Instead, they are methods for you to consider, and hopefully, stimulate you to think of other innovative approaches.

There is not one single approach that is superior to others in all circumstances. To select a method, you should consider the nature of the data, the data structure, the execution time, the memory needed and the easiness to comprehend. Therefore, it is important to know the circumstances and set your priority before deciding on an appropriate approach to duration calculation.

REFERENCES

- [1] SAS Institute Inc. (2004), *SAS 9.1 SQL Procedure User's Guide*. Cary, NC: SAS Institute Inc.
<http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_91/base_sqlproc_6992.pdf> (July 17, 2005)
- [2] SAS Institute (2005). *SAS 9.1.3 Language Reference: Concept*, Second Edition, Cary, NC: SAS Institute Inc.
< http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_lrconcept_8943.pdf > (July 17, 2005)
- [3] SAS Institute Inc. (2005). *SAS 9.1.3 Language Reference: Dictionary*, Third Edition, Cary, NC: SAS Institute Inc.
<http://support.sas.com/documentation/onlinedoc/91pdf/sasdoc_913/base_lrdictionary_9200.pdf> (July 17, 2005)

ACKNOWLEDGMENTS

The author would like to thank her company, Scios Inc., for its support. She is indebted to Ms. Ann Protter, of Scios Inc., for her invaluable comments and to Mr. John Oleynick, of the Cancer Institute of New Jersey, for his suggestions and his thorough work on proofreading this article.

CONTACT INFORMATION

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

Alice Cheng
Scios Inc.
6500 Paseo Padre Parkway
Fremont, CA 94555
Work Phone: (510) 248-2476
Fax: (510) 248-2889
Email: chenga@sciosinc.com

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.

Other brand and product names are trademarks of their respective companies.