SLEEPLESS IN SEATTLE - FOR HOW MANY CONSECUTIVE NIGHTS?
Fikret Karahoda, Merck & Co., Upper Gwynedd, Pennsylvania
Eric Qi, Merck & Company, Inc., Upper Gwynedd, Pennsylvania

ABSTRACT
Looking for the number of consecutive events is an important edit check to find irregularities in otherwise normal data or a key derivation step for important analysis of variables. In this paper, we demonstrate a simple algorithm to find consecutive sleepless events, a common measurement used in Central Nervous System (CNS) studies, in sample data that meets certain criteria.

INTRODUCTION
Study subjects make use of diaries in approximately twenty-five percent of all clinical trials [1]. The diaries are an extremely important tool for capturing data in the Central Nervous System (CNS) therapeutic area. Diaries are used to capture subjects’ day-to-day experiences of events such as total sleep time, no sleep at all on a night, etc. Maintenance of a diary is critical for documenting medical events. As diary data are captured repetitively and consequentially, it is common to look for a pattern of the data using data quality checks or data derivations such as the number of consecutive events. For example, total sleeplessness greater than seven days or no food intake for 15 days may be considered biologically implausible. There are other types of data that require a closer look at consecutive data patterns (e.g., polysomnographic (PSG) data in sleep trials). A PSG may contain an EEG to register brain activity, an EOG to register eye movements, or an EMG to record chin muscle activity during sleep. Sleep stage scoring can be performed visually in 30-second epochs according to Rechtschaffen and Kales criteria [2]. From the PSG, latency to persistent sleep and duration of time measured from lights off to the first epoch of 20 consecutive epochs of non-wake may be derived. This paper describes a simple algorithm to count consecutive events using SAS®.

PROBLEM/TASK
EXAMPLE STUDY DESIGN
In the study design we used, there are three subjects: (PATID =10002, 10004, and 10005). For each subject, data was captured for seven days (OBSR_DT) for a variable named CONDITIONA which may have a value either 1 (true) or 0 (false). We want to display the data for subjects having more than two consecutive days of CONDITIONA = 1. We used the following marked data as an example:

SAMPLE DATA:

```
DATA DATAIN;
format OBSR_DT date9. ;
input @10 PATID  @21 OBSR_DT DATE9. @36 CONDITIONA;
datalines;
10002  09JUN2005          1
10002  10JUN2005          1
10002  11JUN2005          0
10002  12JUN2005          1
10002  13JUN2005          1
10002  14JUN2005          1
10002  15JUN2005          1
10004  27MAY2004          0
10004  28MAY2004          1
10004  29MAY2005          1
10004  30MAY2005          0
10004  31MAY2005          1
```
METHODOLOGY
The first step is to select and order the data that meet specific conditions. In this case, we sorted the data by variables PATID and OBSR_DT and kept only observations satisfying our condition which is CONDITIONA = 1.

PROC SQL;
CREATE TABLE DATAIN_SORTED AS
SELECT * FROM DATAIN
WHERE CONDITIONA = 1
ORDER BY PATID, OBSR_DT;
QUIT;

In the second step, we counted consecutive days that meet specific conditions. This is the key step in this algorithm, and specifically, we define and retain three variables. The first variable is CONSEC_CNT which counts consecutive days that meet the desired condition, CONDITIONA = 1. The second variable is BASE_DT which saves previous date(s), and the third variable is START_DT and it is used to maintain the beginning of the consecutive days. For each new subject, we start with initializing CONSEC_CNT to 1 and set the BASE_DT, START_DT to the value of OBSR_DT. If the difference between BASE_DT and OBSR_DT is 1 (i.e. there is one day difference), OBSR_DT is a consecutive day, and we increase the counter variable CONSEC_CNT by 1. Otherwise, we set the counter variable CONSEC_CNT back to 1. In either case, we move the current date, that is, OBSR_DT, to the BASE_DT. Since we want to display the start date of consecutive events, We set the START_DT to OBSR_DT whenever we reset the counter CONSEC_CNT.

DATA DATAIN_WITH_SEQ_COUNT_ADDED (KEEP= PATID START_DT OBSR_DT CONSEC_CNT);
SET DATAIN_SORTED;
BY PATID OBSR_DT;
RETAIN BASE_DT START_DT CONSEC_CNT;
IF FIRST.PATID THEN DO;
CONSEC_CNT = 1;
BASE_DT = OBSR_DT;
START_DT = OBSR_DT;
END;
IF (OBSR_DT - BASE_DT) = 1 THEN CONSEC_CNT = CONSEC_CNT + 1;
ELSE CONSEC_CNT = 1;
BASE_DT = OBSR_DT;
IF CONSEC_CNT=1 THEN START_DT = OBSR_DT;
FORMAT BASE_DT START_DT DATE9.;
RUN;

In the third and final step, we look for any counter which is greater than our predefined condition; that is greater than or equal to two days.

PROC SQL;
CREATE TABLE OUTDS AS
SELECT PATID, START_DT, OBSR_DT AS END_DT, CONSEC_CNT
FROM DATAIN_WITH_SEQ_COUNT_ADDED
WHERE CONSEC_CNT >= 2
run;
ORDER BY PATID, OBSR_DT;
QUIT;

The output from the last step displays subjects with predefined consecutive days (>= 2) that meet the specific condition (conditiona = 1) is:

<table>
<thead>
<tr>
<th>PATID</th>
<th>START_DT</th>
<th>END_DT</th>
<th>CONSEC_CNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10002</td>
<td>09JUN2005</td>
<td>10JUN2005</td>
<td>2</td>
</tr>
<tr>
<td>10002</td>
<td>12JUN2005</td>
<td>15JUN2005</td>
<td>4</td>
</tr>
<tr>
<td>10004</td>
<td>31MAY2005</td>
<td>01JUN2005</td>
<td>2</td>
</tr>
<tr>
<td>10005</td>
<td>12FEB2006</td>
<td>13FEB2006</td>
<td>2</td>
</tr>
<tr>
<td>10005</td>
<td>18FEB2006</td>
<td>19FEB2006</td>
<td>2</td>
</tr>
</tbody>
</table>

DISCUSSION:
The above SAS program is just one way to derive consecutive day’s data that meet specific conditions. The program uses a simple code to find consecutive records and counting the number of days in a single data step. A paper presented by Pingping Zhang in SUGI29[4] described another algorithm. We created a macro accomplishing the same function as the macro defined in Zhang’s paper. To measure durations of executions, we used an algorithm suggested by Cartney and Hu[6].

We recorded start time, run macro to calculate consecutive days, and finally record the end time. We calculated duration from end and start times. We executed this process 10 times to eliminate probable random effects.

A comparison of two algorithms for an input dataset of 4640 observations is shown in the following table:

<table>
<thead>
<tr>
<th>DURATION (average of 10 runs)</th>
<th>Zhang</th>
<th>Karahoda-Qi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1047 sec</td>
<td>0.0580 sec</td>
<td></td>
</tr>
</tbody>
</table>

These two algorithms accomplished the same function but managed in two different ways. Our one data step approach seems to be more efficient, especially when working with large data.

CONCLUSION:
With this simple algorithm we were able to extract data for further investigation. Diary data may conceal other problems if it is not investigated carefully. By using the method described in this paper, we thoroughly checked and cleaned data that helped us to verify and analyze data more effectively. To explain the algorithm, we used a variable which has date attribute but it can be used with any numeric data having a sequential pattern.

REFERENCES
ACKNOWLEDGMENTS
The authors greatly acknowledge the review and candid feedback from Donna Usavage, Huey Fen Sarah Shiue and Maryann Williams.

TRADEMARKS
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
Fikret Karahoda  
Scientific Programming  
Biostatistics and Research Decision Sciences  
Merck & Co. (UG1D-88)  
North Wales, PA 19454-2505  
Office: (267) 305 1370  
Email: fikret_karahoda@merck.com

Eric Qi  
Scientific Programming  
Biostatistics and Research Decision Sciences  
Merck & Co. (UG1D-88)  
North Wales, PA 19454-2505  
Office: (267) 305 6902  
E-mail: Eric_Qi@merck.com