PharmaSUG 2014 - Paper CC11

Streamline the Dual Antiplatelet Therapy Record Processing in SAS® by Using Concept of Queue and Run-Length Encoding

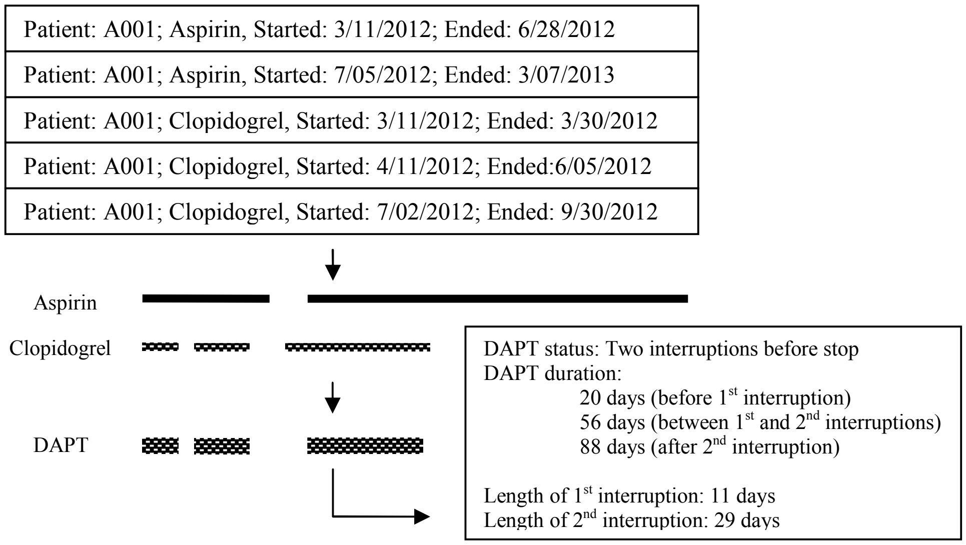
Kai Koo, Abbott Vascular, Santa Clara, CA

ABSTRACT

Dual antiplatelet therapy (DAPT) is a common practice to protect patients from stent thrombosis after stent implantation. In order to analyze its efficiency, medication records of two antiplatelet drugs have to be compiled together. To reduce the coding complexity, the concept of queue is used in SAS data steps. In this approach, multiple records of medication started and stopped status for single patient are rearranged as a “first in, first out (FIFO)” structure through building a new dataset dynamically. Under this queue programming logic, the complete medication record can be derived easily and also compacted into a readable medication history variable simultaneously by using the approach of “run-length data compression”.

INTRODUCTION

Dual antiplatelet therapy, by combining of daily dosages of Aspirin and another antiplatelet medicine (e.g., clopidogrel or ticlopidine, etc.) to protect patients from stent thrombosis, is a common practice after stent implantation. To monitor the safety, efficacy, and optimal duration of DAPT in clinical trials, all related follow-up DAPT medication records have to be analyzed. This analysis involves comparison of multiple medication records from two or more different drugs at a specific time frame (Figure 1). It needs intensive efforts in data cleaning and coding to generate a complete medication history, such as duration of medication and length of interruptions.



Patient: A001; Aspirin, Started: 3/11/2012; Ended: 6/28/2012
Patient: A001; Aspirin, Started: 7/05/2012; Ended: 3/07/2013
Patient: A001; Clopidogrel, Started: 3/11/2012; Ended: 3/30/2012
Patient: A001; Clopidogrel, Started: 4/11/2012; Ended:6/05/2012
Patient: A001; Clopidogrel, Started: 7/02/2012; Ended: 9/30/2012

DAPT status: Two interruptions before stop
DAPT duration:
20 days (before 1st interruption)
56 days (between 1st and 2nd interruptions)
88 days (after 2nd interruption)

Length of 1st interruption: 11 days
Length of 2nd interruption: 29 days

Figure 1. Basic concept of Dual Anti-Platelet Therapy (DAPT) data processing.

Queue is a basic concept of data structure and widely used in computer science in both software development and hardware design. Generally speaking, queue is a set of items (e.g., records) which item can only entered in one end and removed at another end as a “first in, first out (FIFO)” structure. This “FIFO” behavior of queue made it is a nature choice to process data in sequential order, such as the time-ordered DAPT medication records.

To better organize the complete medication history, a Run-Length Encoding (RLE) approach is also demonstrated in this report. RLE is a basic data compression method to store recurrent data as a single data status value and count (Figure 2a). This method is very easy to code and quite efficient to shrink the size of data containing many repeated components without any loss. This simple data encoding technic has been widely used in image and data compression (e.g., Apple® Packbit format and SAS® system option, COMPRESS); this concept is useful in DAPT history summarization as well. Multiple medication records per subject can be compressed into a short and readable string (Figure 2b). This concise data format is not only easy for record reviewing, but also simple in DAPT history analysis coding for further data mining.

(a) Original Data (25 characters) RLE Data (concept)

AAAAAAAAAA BBBBB A BBBB AAAAA → [10 x (A)] [5 x (B)] [1 x (A)] [4 x (B)] [5 x (A)]

(b) RLE in medication records: (Unit: day; 1=on medication, 0=off medication)

111111111111 000 1111111111 → “12, -3, 10” (short string)

Figure 2. Example of RLE algorithm and its implementation

Using the sample DAPT medication records displayed in figure 1 as an example, the medication records of two different medicines can be rearranged to a queue structure based on the order of dates of medication status (Figure 3a). When SAS code reads each observation, the complete DAPT “on” and “off” status are gradually derived (Figure 3b). By the way, the duration of each status is also calculated at the same time by using SAS PROC SORT and DATA steps.

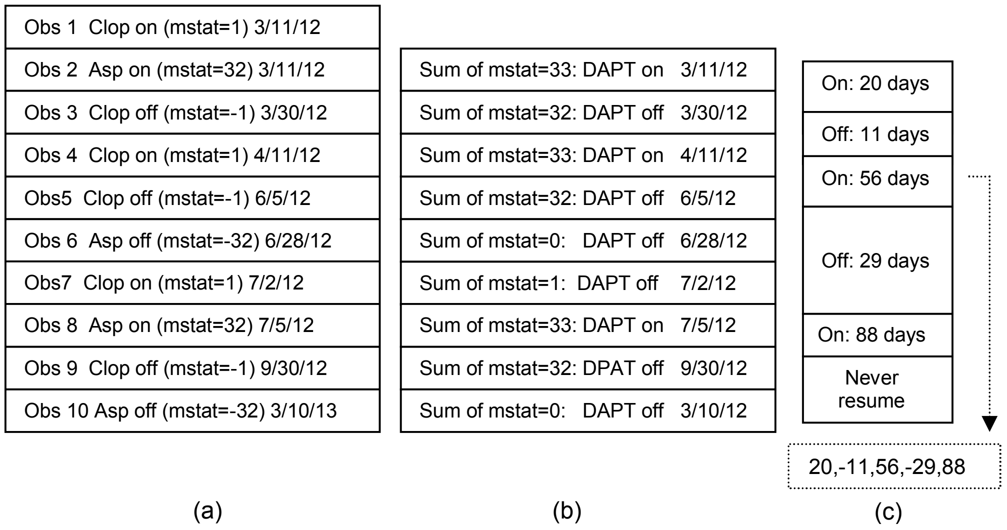

Obs 1 Clop on (mstat=1) 3/11/12
Obs 2 Asp on (mstat=32) 3/11/12
Obs 3 Clop off (mstat=-1) 3/30/12
Obs 4 Clop on (mstat=1) 4/11/12
Obs5 Clop off (mstat=-1) 6/5/12
Obs 6 Asp off (mstat=-32) 6/28/12
Obs7 Clop on (mstat=1) 7/2/12
Obs 8 Asp on (mstat=32) 7/5/12
Obs 9 Clop off (mstat=-1) 9/30/12
Obs 10 Asp off (mstat=-32) 3/10/13
Sum of mstat=33: DAPT on 3/11/12
Sum of mstat=32: DAPT off 3/30/12
Sum of mstat=33: DAPT on 4/11/12
Sum of mstat=32: DAPT off 6/5/12
Sum of mstat=0: DAPT off 6/28/12
Sum of mstat=1: DAPT off 7/2/12
Sum of mstat=33: DAPT on 7/5/12
Sum of mstat=32: DPAT off 9/30/12
Sum of mstat=0: DAPT off 3/10/12
On: 20 days
Off: 11 days
On: 56 days
Off: 29 days
On: 88 days
Never resume
20,-11,56,-29,88
(a)
(b)
(c)

Figure 3. Queue structure (FIFO) of medication records and medication duration calculation. (a) original record queue (b) using variable mstat to determine the medication on/off status (c) result of medication duration calculation.

Through the concepts of queue and RLE, the DAPT data could be handled by SAS efficiently. The complicated medication records will be summarized as a short string (Figure 3c), and free the programmers from the tedious DAPT history analysis coding.

CONCEPTS IMPLEMENTATION

Convert Medication Records to Data Queue. To convert medication records to a queue structure in SAS is quite simple; only PROC SORT and a following short DATA step are necessary. Here is a sample code by using the medication records described in figure 1. This sample code converts medication records of each drug from its original format (Table 1a) to medication record queue (Table 1b) by the order of date of medication status. Basically, only subject ID, date and status of medication are the essential components of this data queue. In this example, an additional variable of procedure date (ProcDt: date of stenting procedure) is also kept as a reference time point for further medication history analysis.

```
%macro build_med_queue(medn);
  proc sort data=med_rec(where=(medcode=&medx)) out=xmed;
    by subjid medstrt;
  run;
  data medq_&mden(keep=subjid mdate mstat procdt);
```

```
    set x_med;
    mstat=1;
    mdate=medstrt;
    output;
    mstat=-1;
    mdate=medend;
    output;
  run;
%mend build_med_queue;
%build_med_queue(c);
%build_med_queue(asp);
```

(a)

	Med_Rec					
obs	Subjid	Medname	Medcode	Medstrt	Medend	Procdt
1	A001	Aspirin	32	3/11/2012	6/28/2012	3/10/2012
2	A001	Aspirin	32	7/05/2012	3/07/2013	3/10/2012
3	A001	Clopidogrel	1	3/11/2012	3/30/2012	3/10/2012
4	A001	Clopidogrel	1	4/11/2012	6/05/2012	3/10/2012
5	A001	Clopidogrel	1	7/02/2012	9/30/2013	3/10/2012

(b)

	Medq_c (Clopidogrel)				Medq_asp (Aspirin)			
obs	Subjid	Mstat	Mdate	Procdt	Subjid	Mstat	Mdate	Procdt
1	A001	1	3/11/2012	3/10/2012	A001	1	3/11/2012	3/10/2012
2	A001	-1	3/30/2012	3/10/2012	A001	-1	6/28/2012	3/10/2012
3	A001	1	4/11/2012	3/10/2012	A001	1	7/05/2012	3/10/2012
4	A001	-1	6/05/2012	3/10/2012	A001	-1	3/07/2013	3/10/2012
5	A001	1	7/02/2012	3/10/2012				
6	A001	-1	9/30/2012	3/10/2012				

Table 1. Sample SAS datasets (a) original medication records, and (b) two medication record queues, Medq_c and Medq_asp, generated by SAS macro %Build_med_queue from Med_rec dataset.

Calculation of Medication Duration and Run-Length Encoding through Queue (1) – Single Medication. The medication data queue presented in Table 1b is ready for medication history processing. The sample code listed below, actually a simple DATA step, is the core section of data processing.

```
%macro Med_History(medx=, mds=);
  data Hist_&medx;
    set &mds;
    by subjid;

    length hist_&medx $80.;
    retain med_d1 med_d2 hist_&medx stat;

    if first.subjid then do;
      med_d1=mdate;
      med_d2=.;
      hist_&medx='';
      stat=.;
    end;

    stat+mstat;

    if stat=0 then do; /* Last day on medication detected */
      if med_d1>procdt then tmp=mdate-med_d1+1;
        else tmp=mdate-med_d1;
      if tmp>0 then hist_&medx=strip(hist_&medx)||','||strip(put(tmp,
        best.));
```

```
          med_d2=mdate+1;
        end;
        else if stat>0 then do; /* First day on medication detected */
          if mdate>med_d2>. then
            hist_&medx=strip(hist_&medx)||','||strip(put(med_d2-mdate,
            best.));
           else if med_d2=. and mdate>procdt+1 then
            hist_&medx=strip(put(procdt-mdate+1, best.));
           med_d1=mdate;
        end;

        if last.subjid then do;
          if index(hist_&medx,',')=1 then hist_&medx =
            substr(hist_&medx,2,length(hist_&medx)-1);
           output;
        end;
     run;
   %mend Med_History;
   %Med_History(medx=1, mds=medq_c);
   %Med_History(medx=32, mds=medq_asp);
```

This DATA step counts medication status (i.e., variable: mstat) by a “FIFO” order. If the value of variable “stat” meets our “on-medication” criteria (i.e., stat=1), it indicate that medication started. At this “on” status, SAS code will calculate the length of medication interruption from previous stop day or procedure date to current date, and retain the “Mdate” value of current observation as a reference point for next medication duration analysis. When the same counter reach 0, it means patient’s last day on medication reached and SAS code will count the duration of medication from previous reference point to current date. By repeating this SAS DATA step, all the durations of “on-medication” or “interruptions” for every patient can be calculated, and the medication history string is built gradually in a “run-length encoding” approach simultaneously.

Subjid	Mstat	Mdate	Procdt	Stat	tmp	Hist_1
A001	1	3/11/2012	3/10/2012	1	.	
A001	-1	3/30/2012	3/10/2012	0	20	,20
A001	1	4/11/2012	3/10/2012	1	.	,20,-11
A001	-1	6/05/2012	3/10/2012	0	56	,20,-11,56
A001	1	7/02/2012	3/10/2012	1	.	,20,-11,56,-26
A001	-1	9/30/2012	3/10/2012	0	88	20,-11,56,-26,91

Calculation of Medication Duration and Run-Length Encoding through Queue (2) – Dual Medication. In the case of DAPT (i.e., two types of drugs), the medication “on” means patient took both Aspirin and another antiplatelet medicine, and the “off” status is defined as any interruption of one of these medications. Although medication records of two types of drugs have to be considered together, the SAS code used to derive the medication history variable for DAPT is still similar to the method described above, except by adding an additional short DATA step (see sample code below) to combine medication record queues of two different drugs into one with the redefined status of DAPT “on” and “off” for further medication history processing.

```
%macro Med_History(medx=, mds=);
  %if &mdex=33 %then %do;
     data &mds
      (drop=mstat rename=(medon=mstat));
       set medq_c medq_asp(in=a);
       by subjid mdate descending mstat;
          retain medon;

       if first.subjid then do;
         medstat=0;
         medon=0;
       end;

      if a then medstat+(mstat*32);
```

Subjid	Mstat	Medsta	Medon	Mdate
A001	1	1	.	3/11/2012
A001*	1	33	1	3/11/2012
A001	-1	32	-1	3/30/2012
A001	1	33	1	4/11/2012
A001	-1	32	-1	6/05/2012
A001*	-1	0	.	6/28/2012
A001	1	1	.	7/2/2012
A001*	1	33	1	7/05/2012
A001	-1	32	-1	9/30/2012
A001*	-1	0	.	3/07/2013

*: Records for Aspirin
Shaded records: final output of combined Medq_dapt dataset.

```
            else medstat+mstat;

          if medstat=&medx then do;
            medon=1;
            output &mds;
          end;
          else if medon=1 and medstat<&medx then do;
            medon=-1;
            output &mds;
          end;
        run;
     %end;

     data Hist_&medx;
        << code segment displayed in previous example >>
     run;
   %mend Med_History;
   %Med_History(medx=1, mds=medq_c);
   %Med_History(medx=32, mds=medq_asp);
   %Med_History(medx=33, mds=medq_dapt);
```

DISSCUSSION AND CONCLUSION

Basically, the queue approach demonstrated in this report relies on a very simple medication status pattern, “on/off/on/off/….”, to derive the history of medication. This approach can track history not only single medicine, but also multiple drugs at the same time. Because, each “on” or “off” status is converted to one SAS data observation, the number of observation number for each subject are assigned dynamically based on the numbers of original records of each patient. It is efficient in computing time, memory allocation and resources usage.

One possible drawback to this queue approach is its low tolerance to data issues. This method would fail, if a correct “on/off/on/off” pattern cannot be generated through original medication records. It is not uncommon that some data entry errors in the month / date of medication start or stop could happen. Using the sample dataset described in figure 1 as an example, if an entry error occurred in the second record of Aspirin that changed the original medication start data from 05JU**L**2012 to 05JU**N**2012, the Aspirin duration derived for this patient will be incorrect.

However, this data-issue sensitive characteristic of queue processing is useful for developing an efficient data checking tool. When the medication data queue with an entry error is further sorted by the date of medication status (sample code and dataset below), the original “on/off/on/off” pattern will be rearranged to “on/on/off/off”. In normal situation, the medication status variable, “stat”, should have only two valid accumulated values: 0 or 1. The presence of any values other than 0 or 1 indicates the existence of data issues such as overlapped / nested medication records or data entry errors (e.g., sample output of issue 1). In this queue approach, those errors in original data can be identified efficiently from a date-ordered data queue using simple SAS PROC SORT and DATA step with a few lines of code.

```
   Proc sort data=medq_asp;
     by subjid mdate;
   run;
   data med_check;
     set medq_asp;
     by subjid;

     mdt=lag(mdate);
     if first.subjid then stat=.;
     stat+mstat;
     if not (stat in (0,1)) or
      (mdt=mdate and stat=1)
      then output;
   run;
```

```
Subjid      Mdate       Mstat   stat
(Issue 1. overlapped duration)
A001      3/11/2012   1     1
A001      6/05/2012   1     2*
A001      6/28/2012  -1     1
A001      3/07/2013  -1     0

(Issue 2. Double daily dosage)
A001      3/07/2012   1     1
A001      3/30/2012  -1     0
A001      3/30/2012*  1     1
A001      4/07/2012  -1     0
Shaded: final output of med_check
for possible data issues
```

Furthermore, some study protocols defined that patient can only take the same medication once a day; any medication record show patient took the same medicine more than once at the same day will be either data entry error or protocol violation. Again, this type of data issue (e.g., sample output of issue 2) is able to be detected quickly by checking the accumulated value of medication status variable: “stat” and the date difference between current observation and previous record from an extra SAS LAG statement based on the medication record queue.

For DAPT study, it is quite common to analyze not only total duration of medication, but also the numbers of interruption and the length of each interruption. In general practice, SAS programmers have to prepare specific variables for different requests in those DAPT analyses; however, the coding is tedious and time consuming. In this report, the RLE-compressed medication history should be one of the solutions to free programmers from complicated coding. Using a simple string or pattern search of minus sign, (“-“), and their related absolute values, the number of interruptions and their durations can be derived at the same time easily. Indeed, this sample code listed below can be re-written to a SAS macro, and many analyses such as, the days from the date of procedure to the first day of each interruption or different medication durations of each DAPT component, can also be calculated from the contents of medication history variables.

```
data chk_interrup(keep=subjid int_cnt int_lgth);
  set Hist_dapt;
  by subjid;

  length d_int $6.;

  if first.subjid then do;
    count=0;
    int_cnt=0;
  end;
  do until(d_int=' ');
    count+1;
    d_int = scan(c_hist, count, ',');
    if index(d_int,'-')> 0 then do;
      int_cnt+1;
      int_lgth=abs(input(d_int, 4.));
      output;
    end;
  end;
run;
```

HIST_DAPT (DAPT history based on data shown in figure 1)

Subjid	Hist_DAPT
A001	20,-11,56,-29,88
...	...

CHK_INTERRUP
(output: DAPT interruption summary)

Subjid	Int_cnt	Int_lgth
A001	1	11
A001	2	29
...	...	

Although there are many approaches to handle DAPT data by using SAS, the queue method with RLE data compression demonstrated in this report is quite straightforward and easy to implement. The code is not only short and resources efficient, but also powerful enough for data cleaning and medication history summary. The final product, comprehensive medication history string, provides further flexibility to deal with many possible detailed analyses of durations and timing of medication status. The concepts described here simplify the coding logic and make complicated DAPT analyses easy to manage.

REFERENCES

- Tenenbaum, Aaron M. and Augenstein, Moshe J. (1981) Queues and Lists, p. 158-216. *In*: Data Structures Using Pascal, Prentice-Hall, Inc.
- Hoyle, Larry (2009) Implementing Stack and Queue Data Structures with SAS® Hash Objects. SUGI 2009, Paper 084-2009.

CONTACT INFORMATION

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

Name: Kai Koo
Enterprise: Abbott Vascular Inc.
Address: 3200 Lakeside Dr.
City, State ZIP: Santa Clara, CA 95054
E-mail: kai.koo@av.abbott.com