PharmaSUG China 2022 - Paper 33

Hash —— One Data Step to Deal with Complex Logic in SDTM/ADaM Generation

Lee Wan, Tigermed Co., Ltd.

ABSTRACT

When coding SDTM/ADaM programs, we always need to generate some complex logics. These logics sometimes require dozens or hundreds of lines of code to complete. This leads to several problems such as difficult to read, difficult to maintain, and slow execution of the code. But if you have some simple knowledge of hash table, many complex code processes can be done with one Data step. This article will illustrate how using hash tables can help reduce the amount of code and improve coding efficiency through code examples and applications, but also need to have a little basic option of hash tables.

INTRODUCTION

SIMPLIFY CODE WITH HASH AND DOW

I will use 3 examples to show how to use Hash and Dow to simplify the code, so that the logic of multiple steps can be completed in one Data step.

MERGE SUPP DOMAIN

In general, the first step for us to program ADaM is to link the SDTM DOMAIN and the corresponding SUPP DOMAIN. Basically, we will use the following two programming ideas to implement.

Method 1:

1. Take a QNAM out of SUPPXX alone.
2. Generate a SAS macro from the above code.
3. Run the above SAS macro in a loop to generate required QNAMs.
4. Merge all the generated datasets with the DOMAIN.

The code for the above steps is shown below:

```
%getsupp(qnam=);
  data &qnam;
    set sdtm.suppxx;
     where qnam="&qnam";
     xxseq=input(idvarval,best.);
     &qnam = qnam;
     keep usubjid xxseq &qnam;
  run;
  proc sort data=&qnam; by usubjid xxseq;run;
%mend;

%getsupp(qnam=xx1);
%getsupp(qnam=xx2);
%getsupp(qnam=xx3);

data adxx;
```

```
  merge sdtm.xx(in=a) xx1 xx2 xx3;
  by usubjid xxseq;
  if a;
run;
```

Method 2:

1. Sort the SUPPXX DOMAIN, then transpose by USUBJID and IDVARVAL, use QNAM as id, use QVAL as var.
2. Check whether there is no QNAM in temporary data, if not, use LENGTH function to dummy, and convert XXSEQ at the same time
3. Because IDVARVAL is a character type, the previous sort cannot be used in the subsequent merge, and it needs to be sorted by XXSEQ
4. Finally, merge the dataset with the SDTM DOMAIN.

The code for the above steps is shown below:

```
proc sort data=sdtm.suppcm out=suppcm;
  by usubjid idvarval;
run;

proc transpose data=suppcm out=suppcm;
  by usubjid idvarval;
  var qval;
  id qnam;
run;

data suppcm;
  length xx $200;/* for dummy*/;
  set suppcm;
  call missing(xx);
  cmseq=input(idvarval,best.);
run;

proc sort data=suppcm;
  by usubjid cmseq;
run;

data adcm;
  merge sdtm.cm(in=a) suppcm;
  by usubjid cmseq;
  if a;
run;
```

It can be seen that the above two methods are very lengthy in terms of code aesthetics and process steps, and are accompanied by multiple sort steps, but if we use Hash and DOW, we can integrate the above multiple steps into one Data step.

The code is shown as below:

```
data adxx;
  if _N_=1 then do;
    dcl hash h(multidata:'Y');
    h.definekey('usubjid', "xxseq");
    h.definedata('qnam', 'qval');
    h.definedone();

    do until(eof);
      set suppxx(keep=usubjid qnam qval idvarval) end=eof;
      xxseq=input(idvarval,best.);
      _N_=h.add();
    end;
  end;

  set sdtm.xx;
  do while (h.do_over()=0);
    if qnam="xx1" then xx1=qval;
      if qnam="xx2" then xx2=qval;
      if qnam="xx3" then xx3=qval;
  end;
  drop qnam qval idvarval;
run;
```

Code's explanation:

1. First, we create a Hash table when _N_=1, then set USUBJID and XXSEQ as Hash keys, and set QNAM and QVAL as Hash data
2. Secondly, use DOW while _N_=1, store the SUPPXX DOMAIN in the Hash table, and convert the value of IDVARVAL at the same time. If you don't need all QNAM, you can use WHERE function to reduce the size of the Hash table.
3. Finally, read SDTM DOMAIN, use the DO_OVER feature of DO WHILE and HASH, traverse the key corresponding to the Hash table, and then use the if statement to extract the required QNAM and QVAL. In this way, even if there is no data in the current SUPPXX, you can dummy the QNAM that you will use.

Is the code very concise? And it runs very efficiently. If you want to add the required QNAM later, you only need to add the relative if statement in the do while.

If we need to add the merge process of ADSL to this code, it is also possible. The upgraded version is shown as follows:

```
data adxx;
  if _N_=1 then do;
    if 0 then set adam.adsl;
    dcl hash h(multidata:'Y');
    h.definekey('usubjid', "xxseq");
    h.definedata('qnam', 'qval');
    h.definedone();
```

```
     dcl hash k(dataset:"adam.adsl");
    k.definekey('usubjid');
    k.definedata(all:'Y');
    k.definedone();

    do until(eof);
      set suppxx(keep=usubjid qnam qval idvarval) end=eof;
      xxseq=input(idvarval,best.);
      _N_=h.add();
    end;
  end;

  set sdtm.xx;
  do while (h.do_over()=0);
    if qnam="xx1" then xx1=qval;
      if qnam="xx2" then xx2=qval;
      if qnam="xx3" then xx3=qval;
  end;
  _N_=k.find();
  drop qnam qval idvarval;
run;
```

DERIVE EPOCH

When generating SDTM, there are many methods to derive EPOCH variable. The difficulty is that you must link SE and dataset in a many-to-many manner, and then according to the date range, the desired epoch is obtained in advance, but sometimes the date will match 2 epochs, and further deletion is required, but if it is not handled properly, the number of entries will increase or decrease. Here we still use hash to solve this problem, the code is as follows

```
data Outdata;
  if _n_=1 then do;
    if 0 then set sdtm.se(keep=usubjid sestdtc seendtc epoch);
    dcl hash h(dataset:'sdtm.se',multidata:'Y');
    h.definekey('usubjid');
    h.definedata('sestdtc','seendtc','epoch');
    h.definedone();
      call missing(sestdtc,seendtc,epoch);
  end;

  set indata;
  do while(h.do_over()=0);
      if ''<sestdtc<=indatc<=seendtc then e=epoch;
  end;
  epoch=e;
  drop indatc sestdtc seendtc e;
run;
```

First, we put SE DOMAIN into the Hash table, and put USUBJID as the Hash key. Because one USUBJID has multiple EPOCHS for SE DOMAIN, we need to set MULTIDATA as Y, and then store SESTDTC, SEENDTC and EPOCH into Hash data.

Then using Do While and DO_OVER to traverse, and match EPOCH based on the date range. Here, you can adjust greater than or less than to achieve whether the date takes the EPOCH backward or the

EPOCH forward. At the same time, because the SET function is used, the observation of dataset will not be deleted and added.

DERIVER BORC

As we all know, BORC's logic is more complicated, because its logic requires 2 pieces of data to confirm the first piece of data, and SAS processes data line by line, so it is difficult to use basic methods to program. But we can traverse the data well by storing the data in the Hash table. Since the code to derive all BORCs is too long, let's take a look at confirm CR. The logic of confirm CR is that 2 CRs appear in a row, and the middle of the two CRs except for NE, no other response value can appear, and the CR that appears for the first time is the confirm CR.

The code as follows:

```
    data _null_;
        if _n_=1 then do;
          dcl hash h(multidata : 'Y');
          h.definekey("usubjid");
          h.definedata("_HDate_","_HResp_");
          h.definedone();
          dcl hiter hi ('h');

          dcl hash k(dataset:"_borc1_(obs=0)",ordered:'Y');
          k.definekey("usubjid");
          k.definedata(all:'Y');
          k.definedone();
        end;



        *--------------------;
        * Get first pd date ;
        *--------------------;
        do _N_=1 by 1 until(last.usubjid);
          set _borc1_;
          by usubjid;
          if (_rsPDdt_ =. or _rsPDdt_>_rsADT_) and _rsResp_='PD' then
_rsPDdt_=_rsADT_;
        end;

        *-----------------------------------;
        * Get response before first pd date ;
        *-----------------------------------;
        do until(last.usubjid);
          set _borc1_;
          by usubjid;
          _HDate_ = _rsADT_;
          _HResp_ = _rsResp_;
          if (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then h.add();
        end;

        do until(last.usubjid);
          set _borc1_ end=eof;
          by usubjid;
          *--------------------;
          * CR confirm        ;
          *--------------------;
```

```
          if _rsResp_='CR' and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then do;
            cr=1;
            do rc=hi.first() by 0 while(rc=0);
              if _HDate_>_rsADT_ then do;
                if _HResp_ = 'CR' then cr=cr+1;
                else if _HResp_ = 'NE' then cr=cr*1;
                else cr=-999;
              end;
              if _HDate_-_rsADT_>=&CRwin and _HResp_ = 'CR' and cr>1 and
k.check() then k.add();
              rc=hi.next();
            end;
            put usubjid= cr=;
          end;
        end;
        h.clear();

        if eof then do;
          k.output(dataset:"_borc2_");
        end;
    run;
```

First block of code:

```
        if _n_=1 then do;
          dcl hash h(multidata : 'Y');
          h.definekey("usubjid");
          h.definedata("_HDate_","_HResp_");
          h.definedone();
          dcl hiter hi ('h');

          dcl hash k(dataset:"_borc1_(obs=0)",ordered:'Y');
          k.definekey("usubjid");
          k.definedata(all:'Y');
          k.definedone();
        end;
```

Here we have created 2 Hash tables, one is to store the data of RS, and the other is to store the result of BORC. For the Hash table ‘h’, we put USUBJID as the Hash key, and Hash data stores the date and response, and we additionally create an iterator ‘hi’ over this Hash table.

Second block of code:

```
        *-------------------;
        * Get first pd date ;
        *-------------------;
        do _N_=1 by 1 until(last.usubjid);
          set _borc1_;
          by usubjid;
          if (_rsPDdt_ =. or _rsPDdt_>_rsADT_) and _rsResp_='PD' then
_rsPDdt_=_rsADT_;
        end;
```

We use DOW to read the RS data and get the date of the earliest PD of each USUBJID, which is to filter the data before PD later.

Third block of code:

```
        *-----------------------------------;
        * Get response before first pd date ;
        *-----------------------------------;
        do until(last.usubjid);
          set _borc1_;
          by usubjid;
          _HDate_ = _rsADT_;
          _HResp_ = _rsResp_;
          if (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then h.add();
        end;
```

Here we still use DOW to read the RS data, and rename the variables of ADT and RESP of RS to the 'h' Hash table, and use the first PD date obtained by the previous DOW to perform a screening.

Forth block of code:

```
        do until(last.usubjid);
          set _borc1_ end=eof;
          by usubjid;
          *-------------------;
          * CR confirm        ;
          *-------------------;
          if _rsResp_='CR' and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then do;
            cr=1;
            do rc=hi.first() by 0 while(rc=0);
              if _HDate_>_rsADT_ then do;
                if _HResp_ = 'CR' then cr=cr+1;
                else if _HResp_ = 'NE' then cr=cr*1;
                else cr=-999;
              end;
              if _HDate_-_rsADT_>=&CRwin and _HResp_ = 'CR' and cr>1 and
k.check() then k.add();
              rc=hi.next();
            end;
            put usubjid= cr=;
          end;
        end;
        h.clear();
```

It is more complicated here. It is also used to read the data in DOW, and then make a preliminary judgment on the data to determine whether it is a CR, and before the first PD. Then set a counter 'cr' and initialize the value to 1. Then use the iterator to traverse the data after the CR in order by USUBJID. If a CR is found in the traversal, the counter 'cr' is increased by 1, and if it is NE, it is multiplied by 1, that is, the value of CR will not change, if it is another value, it will be directly assigned -999. After traversing one, it is directly judged. If the counter 'cr'>1, it is added to the Hash table named k. In order to avoid multiple CRs being stored, CHECK() is used here to check whether there is a confirm CR and then store it.

fifth block of code:

```
if eof then do;
          k.output(dataset:"_borc2_");
        end;
```

Finally, output the Hash table named k.

The complete code as below:

```
data _null_;
    if _n_=1 then do;
      dcl hash h(multidata : 'Y');
      h.definekey("usubjid");
      h.definedata("_HDate_","_HResp_");
      h.definedone();
      dcl hiter hi ('h');

      dcl hash k(dataset:"_borc1_(obs=0)",ordered:'Y');
      k.definekey("usubjid");
      k.definedata(all:'Y');
      k.definedone();
    end;

    *-------------------------------------------------------------------------
--;
    * Get first pd
date                                                          ;
    *-------------------------------------------------------------------------
--;
    do _N_=1 by 1 until(last.usubjid);
      set _borc1_;
      by usubjid;
      if (_rsPDdt_ =. or _rsPDdt_>_rsADT_) and _rsResp_='PD' then
_rsPDdt_=_rsADT_;
    end;

    *-------------------------------------------------------------------------
--;
    * Get response before first pd
date                                          ;
    *-------------------------------------------------------------------------
--;
    do until(last.usubjid);
      set _borc1_;
      by usubjid;
      _HDate_ = _rsADT_;
      _HResp_ = _rsResp_;
      if (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then h.add();
    end;

    do until(last.usubjid);
      set _borc1_ end=eof;
      by usubjid;
      *-------------------------------------------------------------------------
----;
      * CR
```

```
confirm                                                                 ;
      *-------------------------------------------------------------------------
----;
      if _rsResp_='CR' and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then do;
        cr=1;
        do rc=hi.first() by 0 while(rc=0);
          if _HDate_>_rsADT_ then do;
            if _HResp_ = 'CR' then cr=cr+1;
            else if _HResp_ = 'NE' then cr=cr*1;
            else cr=-999;
          end;
          if _HDate_-_rsADT_>=&CRwin and _HResp_ = 'CR' and cr>1 and
k.check() then k.add();
          rc=hi.next();
        end;
        put usubjid= cr=;
      end;
    end;

    do until(last.usubjid);
      set _borc1_ end=eof;
      by usubjid;
      *PR confirm;
      if _rsResp_='PR' and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) then do;
        pr=1;
        do rc=hi.first() by 0 while(rc=0);
          if _HDate_>_rsADT_ then do;
            if _HResp_ in ('CR' 'PR') then pr=pr+1;
            else if _HResp_ = 'NE' then pr=pr*1;
            else pr=-999;
          end;
          if _HDate_-_rsADT_>=&CRwin and _HResp_ in ('CR' 'PR') and pr>1 and
k.check() then k.add();
          rc=hi.next();
        end;
        put usubjid= pr=;
      end;
    end;

    *SD confirm;
    do until(last.usubjid);
      set _borc1_ end=eof;
      by usubjid;
      if _rsADT_-&rfstdt.+1>=&SDwin. and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.)
then do;
        if k.check() and _rsResp_ in ('CR' 'PR' 'SD') then do;
          _rsResp_='SD';
          k.add();
        end;
    /*  if k.find()=0 and rsstresc in ('COMPLETE RESPONSE' 'PARTIAL RESPONSE'
'STABLE DISEASE') and rsdy<date then do;*/
    /*    rsstresc='STABLE DISEASE';*/
    /*      k.replace();*/
    /*  end;*/
      end;
    end;
```

```
    *PD confirm;
    do until(last.usubjid);
      set _borc1_ end=eof;
      by usubjid;
      *CR->PD or PR->PD or SD->PD and <=sd window;
      if _rsADT_-&rfstdt.+1<&SDwin. and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) and
k.check() then do;
        if _rsResp_ in ("CR" "PR" "SD") then do;
          fpd=1;
          do rc=hi.first() by 0 while(rc=0);
            if _HDate_>_rsADT_ then fpd=fpd+1;
            if _HDate_>_rsADT_ and _HResp_="PD" and fpd=2 then do;
              _rsResp_="PD";
              _rsADT_=_rsPDdt_;
              k.add();
            end;
            rc=hi.next();
          end;
        end;
      end;

      *Only PD or NE;
      if _rsResp_="PD" and k.check() then do;
        pd=1;
        do rc=hi.first() by 0 while(rc=0);
          if _HResp_ not in ("NE" "PD") then pd=pd*0;
          rc=hi.next();
        end;
        if pd=1 then k.add();
      end;
      put usubjid= pd= fpd=;
    end;

    *NE confirm;
    do until(last.usubjid);
      set _borc1_ end=eof;
      by usubjid;
      if _rsResp_='NE' and k.check() then k.add();
      if _rsADT_-&rfstdt.+1<&SDwin. and (_rsADT_<=_rsPDdt_ or _rsPDdt_=.) and
k.check() and _rsResp_ in ("CR" "PR" "SD") then do;
        _rsResp_='NE';
        k.add();
      end;
    end;

    h.clear();

    if eof then do;
    do until(eom);
      set adam.adsl(keep=usubjid &rfstdt.) end=eom;
      where not missing(&rfstdt.);
      if k.check() then do;
        _rsResp_='NA';
        call missing(rsdtc,rsseq);
        k.add();
      end;
    end;
```

```
    k.output(dataset:"_borc2_");
    end;
run;
```

CONCLUSION

Hash table and DOW can also be used for many complex logics such as derive visit. The combination of the two is very beneficial to our daily programming. You can dig a lot to make our code more briefly and efficiently.

REFERENCES

- The DOW-Loop Unrolled. Available at http://support.sas.com/resources/papers/proceedings09/038-2009.pdf
- Data Management Solutions Using SAS Hash Table Operations. Available at https://sasinstitute.redshelf.com/book/1878343

CONTACT INFORMATION

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

Lee Wan
Tigermed Co., Ltd.
lee.wan@tigermedgrp.com