

The Power of RETAIN Statement in Clinical Trial Programming by Examples

Baoxian Lan, Everest Clinical Research Services Inc., Toronto, Canada
Daniel Tsui, Everest Clinical Research Services Inc., Toronto, Canada
Shifang Liu, Everest Clinical Research Services Inc., Toronto, Canada

ABSTRACT

RETAIN statement is a commonly used statement in clinical trial programming. This paper will demonstrate the power of using RETAIN statement in the following daily programming work: 1) calculating change from baseline; 2) LOCF (Last Observation Carried Forward); 3) transposing data; 4) Re-ordering variable order; 5) checking sequential date order. Examples will be given for each technique. The paper will also explain how inexplicit RETAIN works. Caution on using RETAIN statement will be mentioned.

INTRODUCTION

In clinical trial programming, tasks like calculating change from baseline, using LOCF method to impute missing values, transposing data, calculating total score, etc. are very common. We deal with these tasks almost every single day. Even though you can use different techniques to accomplish these tasks, using RETAIN statement to do them makes it simple and easy, usually in one data step. This paper will pick up a couple examples from our daily work to share with you and show you how these things can be done quickly and efficiently by using RETAIN statement.

EXAMPLE 1: CALCULATING CHANGE FROM BASELINE

Calculating change from baseline is one of the most common tasks in daily clinical trial programming. As mentioned earlier, you can do it in different ways. But you can use the RETAIN statement to retain baseline value and then calculate the change from baseline as shown in the following example. One thing you have to remember here is to set the baseline value to missing (or zero depending on your data) at first.patient (byvar). Otherwise, the baseline value will be carried to the next patient.

```
data change;
  set score;
  by inv patient visitno;

  retain baseline;

  * set to missing to avoid being carried to the next patient;
  if first.patient then baseline=.;

  if visitno=1 then baseline=score;

  if last.patient then do;
    if visitno gt 1 then do;
      endpoint=score;
      if baseline ne . and endpoint ne . then do;
        change=endpoint - baseline;
        pctchg=change/baseline*100;
      end;
    end;
    output;
  end;

  keep inv patient visitno center treatment baseline endpoint change pctchg;
run;
```

EXAMPLE 2: LOCF

Even though clinical trial is normally well controlled, there are always missing visits or values. So using LOCF method to impute missing values is also a very common task in clinical trial programming. Here is a simple example to demonstrate how to use RETAIN statement to carry the non-missing values forward.

```
* locf total score;
data hamdlocf;
  set totalscore;
  where visitno not in (1,2);
  by inv patient visitno;

  retain ttlLOCF;

  * set to missing to avoid being carried to the next patient;
  if first.patient then ttlLOCF=.;

  if ttlscore ne . then ttlLOCF=ttlscore;

  keep inv patient visitno ttlscore ttllocf;
run;
```

EXAMPLE 3: TRANSPOSING DATA - MANY TO ONE

We often need to transpose data from one observation per visit to one observation per patient and calculate total score/value at the same time. You can use proc transpose to do this. But you would need at least two data steps to transpose the data. However, using RETAIN statement, you can do it in just one data step. The trick here is to use visit instead of i to skip missing visits. Otherwise, if a visit is missing, the results will be wrong (e.g. v2 has visit 3 value).

```
data pat_score;
  set v_score;
  by patient visit;

  array v(5) v1-v5;

  retain v1-v5;

  * set to missing to avoid being carried to the next patient;
  if first.patient then do i=1 to 5;
    v(i)=.;
    pat_score=.;
  end;

  v(visit)=sum_score;

  pat_score+sum_score;

  if last.patient then output;

  drop i;
run;
```

EXAMPLE 4: RE-ORDERING VARIABLE ORDER

Sometimes, we are required to re-order variables in a dataset in a certain order, especially after we create new variables or merge different datasets to create new dataset. This can be done by giving a RETAIN statement before SET statement or MERGE statement in a data step.

```
data FINL_ANL;
  retain prot site_no inv_no pat_no case_id cntrcode age sex race trtmnt smdur
         visitn fin_dta;

  merge KEYVARS PLATE021(in=i);
  by numsub;
  if i;

run;
```

EXAMPLE 5: CHECKING SEQUENTIAL DATE ORDER

We are usually asked to help data management to do some cross checking or logic checking in data cleaning. Checking sequential date order could be one of these checks. Here is an example we have done to do the check using RETAIN statement. A little trick here is to output data before assigning the previous value and retaining it to the next observation.

```
%macro e11_6b;
  %let msg1=If 'Date of Assessment' and 'Time' are present, they should be;
  %let msg2=greater than the previous sequence.;
  %let msg3=i.e. T42 > T36 > T30 > T24 > T18 > T12 > T10 > T8 > T6 > T4 > T2.;

  proc sort data=PLATE012;
    by subno dfseq;
  run;

  data e11_6b &keep2;
    set plate012;
    by subno dfseq;

    assdate=input(dtvis,?? yymmdd10.);
    asstime=hms(hhpain,mipain,0);

    retain prvdate prvtime;

    * set to missing to avoid being carried to the next patient;
    if first.subno then do;
      prvdate=.;
      prvtime=.;
    end;

    if . lt assdate lt prvdate or
      (assdate=prvdate and (. lt asstime lt prvtime)) then do;

      &length;
      edit = "11.6b";
      msg = "&msg1 &msg2 &msg3";
      var = "date,time";
      value = dtvis||","||put(asstime,time5.);

      output;

      prvdate=assdate;
      prvtime=asstime;

    end;
  run;
%mend e11_6b;
```

INEXPLICIT RETAIN

All the examples above are mainly to show explicit retain. The following example is to show inexplicit retain.

```
data check;
  set compliance;
  by inv patient visitno;

  retain prevcom;

  if first.patient then do;
    prevcom=.;
    problem=.;
  end;

  if prevcom gt 30 and compliance gt 30 then problem + 1; * inexplicit retain;

  output;
```

PhUSE 2006

```
prevcom=compliance;

run;
```

Inexplicit retain is often used in summing total scores / values. The neat thing to use inexplicit retain is that only non-missing values will be retained even though there is missing value in the data, if that's what you want. For example,

```
data sum_score;
  set score;
  by patient visit;

  if first.patient then sum_score=.;

  sum_score + score;

  if last.patient;
run;
```

Inexplicit retain above is equal to the following explicit retain.

```
data sum_score;
  set score;
  by patient visit;

  retain sum_score;

  if first.patient then sum_score=.;

  sum_score=sum_score + score;

  if last.patient;
run;
```

But if you forget to use RETAIN statement in the above example and there is missing visit value, then total score will be missing. See example below.

```
data sum_score;
  set score;
  by patient visit;

  if first.patient then sum_score=.;

  sum_score=sum_score + score;

  if last.patient;
run;
```

CAUTION ON USE OF RETAIN STATEMENT WITH WHERE CLAUSE AND IF CONDITION

The following example did not work when visit=8 and prevdose=., i.e. the check won't trigger even though there is non-compliance.

```
proc sort data=PLATE012 out=SM;
  by patnum visit;
run;

data SM;
  set SM;
  by patnum visit;

  retain prevdose;

  if first.patnum and visit=4 then prevdose=.;

  if visit > 4 and smccply=2 and prevdose ne smdostkn;

  &length;
  edit ="9.9";
```

PhUSE 2006

```
msg  ="&msg1 &msg2";
var  ="smccply";
value=trim(left(put(smccply, ynfmt.)) );

output;
prevdose=smdosdi;
run;
```

The solution is to split the above data step into two steps as follow.

```
proc sort data=PLATE012 out=SM;
  by patnum visit;
run;

data SM;
  set SM;
  by patnum visit;

  retain prevdose;

  if first.patnum and visit=4 then prevdose=.;

  output;
  prevdose=smdosdi;
run;

data be9_9 &keep2;
  set SM;
  by patnum visit;
  where visit > 4;

  sumdose=sum (smdostkn,smdosunu,smdoslos);

  if smccply=2 and prevdose ne sumdose;
  &length;
  edit ="9.09";
  msg  ="&msg1 &msg2";
  var  ="smccply";
  value=trim(left(put(smccply, ynfmt.)) );
run;
```

CONCLUSION

We have used RETAIN statement extensively in our daily programming work. We found that RETAIN statement is simple and easy to use. If you use it skillfully, it will make your programming work more efficient and your life easier. But remember to check every single time to see if your code works.

CONTACT INFORMATION

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

Baoxian Lan
Everest Clinical Research Services Inc.
175 Commerce Valley Drive West, Suite 230
Toronto, Ontario L3T 7P6, Canada
Work Phone: (905) 695 3741
Fax: (905) 695 3757
Email: bob.lan@ecrscorp.com
Web: www.ecrscorp.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.