Paper P01

One-Step Change from Baseline Calculations, and Other DOW-Loop Tricks

Nancy Brucken, i3 Statprobe, Ann Arbor, MI

ABSTRACT

Change from baseline is a common measure of safety and/or efficacy in clinical trials. The traditional way of calculating changes from baseline in a vertically-structured dataset requires multiple DATA steps, and thus several passes through the data. This paper demonstrates how change from baseline calculations can be performed with a single pass through the data, through use of the Dorfman-Whitlock DO- (DOW-) Loop. It also looks at how the DOW-Loop can be implemented to transpose multiple variables onto a single record per BY statement in one pass through the input dataset, thus by-passing one limitation of PROC TRANSPOSE.

BACKGROUND

The most common algorithm for computing change from baseline involves first physically dividing the original dataset into baseline and post-baseline datasets. The baseline value may then be obtained from a single point in time, often from the last value measured before the first dose of study drug, or may be a composite of several observations- perhaps the mean of the last two measurements before the start of study drug administration. A dataset containing the baseline value for every subject is created, and then merged with the post-baseline dataset by subject. Every post-baseline record thus contains the baseline value for that subject, and the change from baseline to that visit record is calculated.

So, given a dataset that looks like this:

```
USUBJID    VISITC       VISITN    HR

    1      Screening       1      91
    1      Day 1           2       .
    1      Week 1          3      68
    1      Week 2          4      73
    1      Week 4          5      96

    2      Screening       1       .
    2      Day 1           2      73
    2      Week 1          3      73
    2      Week 2          4      52
    2      Week 4          5      59
```

Traditional code for computing change from baseline would go something like this:

```
*** Separate baseline and post-baseline values;
data baseline postbase;
  set save.vitals;
  if visitn <= 2 then output baseline;
    else output postbase;
run;

*** Baseline value is last non-missing value before first dose;
data baseline1 (keep=usubjid hr rename=(hr=bl));
  set baseline (where=(hr is not missing));
  by usubjid visitn;
  if last.usubjid;
run;

*** Combine with post-baseline data and calculate change from baseline;
data postbase1;
  merge postbase (in=inp) baseline1 (in=inb);
  by usubjid;
  chgbl = hr - bl;
run;
```

And the final dataset of post-baseline values looks something like this:

```
USUBJID    VISITC    VISITN    HR    BL    CHGBL

    1      Week 1       3      68    91     -23
    1      Week 2       4      73    91     -18
    1      Week 4       5      96    91       5

    2      Week 1       3      73    73       0
    2      Week 2       4      52    73     -21
    2      Week 4       5      59    73     -14
```

Note that we have made three passes through the data- the first to split the dataset, the second to process the baseline values separately, and the third to actually compute changes from baseline. Granted, the last two datasets combined form the original dataset, so in reality, we've really made only two passes through the complete data. But still, for a study with thousands of subjects, processing a dataset with many parameters such as clinical laboratory data can still take awhile.

THE DOW-LOOP

The DOW-Loop is a technique originally developed by Don Henderson, and popularized on the SAS-L listserv by Paul Dorfman and Ian Whitlock. It involves taking control of the implicit DO-Loop inherent in the DATA step, identifying and storing the baseline value in a variable that is retained until all post-baseline values for a subject have been processed, and then writing out only those post-baseline values.

The DOW-Loop relies on the fact that the values of assigned variables, or variables created by assignment statements in the DATA step, are not reset to missing until SAS returns to the top of the DATA step. In the following example, it processes all of the records for a given subject first, returning to the top of the step only after the last record for that subject has been handled.

The following code makes use of the DOW-Loop to compute change from baseline and generate a dataset containing change from baseline for all post-baseline measurements in a single DATA step requiring only one pass through the data:

```
data postbase;

  do until (last.usubjid);
    *** Only keep non-missing pre-dose values;
    set save.vitals (where=(not(visitn <= 2 and hr is missing)));
      by usubjid visitn;

      if visitn <= 2 then bl = hr;
        else do;
          chgbl = hr - bl;
          output;
        end;
  end;

run;
```

And the resulting dataset is identical to the one obtained through traditional means:

```
USUBJID     VISITC     VISITN     HR     BL     CHGBL

    1       Week 1        3       68     91      -23
    1       Week 2        4       73     91      -18
    1       Week 4        5       96     91        5

    2       Week 1        3       73     73        0
    2       Week 2        4       52     73      -21
    2       Week 4        5       59     73      -14
```

Let's take a closer look at what is going on inside of the Program Data Vector (PDV).

INTERNAL PROCESSING DETAILS

The first thing SAS does when it encounters a DATA step is to compile the statements in the step, and set up the PDV. Once the code is compiled, it is then executed. At the start of the DATA step, all computed variables are set to missing. The PDV looks something like this:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
1	.	.	.	.	1	1	1	.	.

SAS then immediately starts into the DO-loop. The UNTIL condition is not evaluated until the END statement at the bottom of the loop, so SAS proceeds on to the SET statement, and reads in the first record in the dataset. The PDV changes to this:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
0	1	Screening	1	91	1	1	1	.	.

VISITN=1, so the condition IF VISITN <= 2 is met, and the PDV changes to:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
0	1	Screening	1	91	1	1	1	91	.

SAS then returns to the top of the DO-loop, not the top of the DATA step; thus, the value of BL is retained, since it is an assigned variable. The second record in the dataset does not meet the WHERE condition, and is discarded. When the third record is read in, the PDV looks like:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
0	1	Week 1	3	68	0	1	1	91	.

VISITN=3, so the condition IF VISITN <= 2 is not met. Control passes to the ELSE branch of the IF statement, change from baseline is calculated, and the record is output. The PDV looks like:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
0	1	Week 1	3	68	0	1	1	91	-23

SAS returns to the top of the DO-loop again, and repeats the process until the last record for this subject has been processed. Once that happens, it finally goes back to the top of the DATA step, and the PDV looks like this:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
1	**1**	**Week 4**	**5**	**59**	**0**	**1**	**1**	**.**	**.**

Note that the values displayed in red, which come from variables either created by SAS or read in via the SET statement, have not yet been overwritten, since the next SET statement has not been executed. However, the values for BL and CHGBL have been set back to missing, since they are assigned variables.

The DO-loop once again takes control, and the next record is read in. However, it does not meet the conditions required by the WHERE clause, and so the third record is read in:

LAST. USUBJID	USUBJID	VISITC	VISITN	HR	FIRST. USUBJID	FIRST. VISITN	LAST. VISITN	BL	CHGBL
0	**2**	**Day 1**	**2**	**73**	**1**	**1**	**1**	**.**	**.**

The values displayed in red have now all been overwritten by those on the new record read in via the SET statement. The process is then repeated for the remaining records in the input dataset.

PROBLEM 2- MULTI-COLUMN TRANSPOSE

Suppose we have a SAS dataset of summarized laboratory data, containing one record per parameter, category and treatment group. We need to report that data in a table with one record per parameter and category, and the treatment group counts displayed in separate columns. The input dataset looks like this:

ANALYTEC	LISTTYPE	LISTNUM	RXGRP	PATCT	DENOM
CALCIUM	Abnormal Baseline Values	1	1	6	260
CALCIUM	Abnormal Baseline Values	1	2	4	227
CALCIUM	Normal Baseline and Abnormal EOT Values	2	1	5	260
CALCIUM	Normal Baseline and Abnormal EOT Values	2	2	6	227

We need to transpose this data to a reporting dataset looking like this:

ANALYTEC	LISTTYPE	LISTNUM	RXGRP	PATCT1	PATCT2	DENOM1	DENOM2
CALCIUM	Abnormal Baseline Values	1	1	6	5	260	227
CALCIUM	Normal Baseline and Abnormal EOT Values	2	2	4	6	260	227

One possible solution is to use PROC TRANSPOSE to transpose the dataset by parameter and category. However, the following code, transposing both variables in the same step, yields the dataset shown below, which is not quite what we are looking for.

```
PROC TRANSPOSE DATA=LABS OUT=LABTRAN;
  BY ANALYTEC LISTTYPE LISTNUM;
  VAR PATCT DENOM;
  ID RXGRP;
RUN;
```

yields:

ANALYTEC	LISTTYPE	LISTNUM	_NAME_	_1	_2
CALCIUM	Abnormal Baseline Values	1	PATCT	6	5
CALCIUM	Abnormal Baseline Values	1	DENOM	260	227
CALCIUM	Normal Baseline and Abnormal EOT Values	2	PATCT	4	6
CALCIUM	Normal Baseline and Abnormal EOT Values	2	DENOM	260	227

We can then split this dataset apart so that the PATCT and DENOM records end up stored in separate datasets, and merge it back together by parameter and category, in order to reach our target dataset. Note, though, that PROC TRANSPOSE requires one pass through the entire dataset, splitting the resulting dataset apart requires another pass, and the subsequent merge makes still another pass through the dataset. For small datasets, the three passes through the data may not take much time or system resources. However, for larger datasets, each pass may take a significant amount of execution time and/or resources.

ALTERNATIVE SOLUTION

An alternative solution involves a single pass through the dataset using a DOW-Loop. This powerful technique moves the DATA step SET statement inside of a explicitly-coded DO-loop, thus giving the programmer complete control over retention of variable values and the population of the Program Data Vector (PDV). Remember that values of assigned variables, or variables created by assignment statements in the DATA step, are not reset to missing until SAS returns to the top of the DATA step.

The code to generate the desired reporting dataset directly from the input dataset looks like:

```
data labtran (drop=i patct denom rxgrp);
  array patcts (*) patct1-patct2;
  array denoms (*) denom1-denom2;

  *** Initialize arrays;
  do i=1 to dim(patcts;
    patcts(i) = 0;
    denoms(i) = 0;
  end;

  do until (last.listnum or eof);
    set labs end=eof;
    by analytec listnum;

    *** Populate arrays;
    patcts(rxgrp) = patct;
    denoms(rxgrp) = denom;
  end;

  output;
run;
```

The ARRAY statements define two arrays, one in which to store the patient subgroup counts for each treatment group, and one in which to store the total number of patients in each treatment group. The array initialization loop is executed every time control returns to the top of the DATA step, after the last record for a category has been output.

The DOW-Loop itself begins with the DO UNTIL statement, and takes control from the traditional implicit DATA step loop. Because the SET statement is inside of the DOW-loop, the loop is not exited until after the last record for the category has been processed, and the array elements populated inside of the loop are retained while all of the records for the category are read in, The array elements are assigned variables in this example, and so are not reset until SAS reaches the top of the DATA step. Thus, we can store values in PATCT1, PATCT2, DENOM1 and DENOM2 for all of the records in the category during the execution of the DOW-loop, without having them reinitialized at the top of the DATA step. The single record for the category, containing the filled arrays, is output at the completion of the DOW-loop. Control then returns to the top of the DATA step, the arrays are reinitialized, and the records for the next category are read in.

CONCLUSION

The DOW-Loop technique takes advantage of the fact that SAS does not reset the values of assigned variables until it reaches the top of the DATA step. Coding an explicit DO-loop preventing SAS from returning to the top of the DATA step until it has read in all of the records for a given subject allows you to retain a baseline value through processing of all post-baseline records for that subject, and automatically reinitializes the computed baseline and change from baseline variables after the last record for that subject has been processed. This technique then enables the calculation of change from baseline in a single DATA step, instead of the multiple passes through the data required by the algorithm frequently applied to this problem. It can reduce the number of passes through a dataset required by a program in other applications, such as multi-column transposes, and fewer passes through the dataset generally result in faster and more efficient programs.

REFERENCES

For information on how the PDV is populated, see the SAS 9.1.3 On-Line Help and Documentation.

For more information on applications of the DOW-Loop, visit the SAS-L listserv archives at http://www.listserv.uga.edu/archives/sas-l.html, and search for postings by Ian Whitlock and Paul Dorfman, among others.

Dorfman, Paul. (2008) “The DOW-Loop Unrolled”. PharmaSUG 2008 Conference Proceedings, housed at http://www.lexjansen.com.

Chakravarthy, Venky, (2005). “RETAIN or NOT? Is LAG Far Behind?”. PharmaSUG 2005 Conference Proceedings, housed at http://www.lexjansen.com.

ACKNOWLEDGMENTS

Thanks to my colleagues at i3 Statprobe, who reviewed this paper and provided helpful suggestions, to all of the people who gave me the inspiration for using this technique through their postings on SAS-L, and to Don Henderson, who, so far as anyone can determine, was probably the first person to try it..

CONTACT INFORMATION

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

Nancy Brucken
i3 Statprobe
300 West Morgan Rd.
Ann Arbor, MI 48108

Work Phone: (734) 757-9045
E- mail: Nancy.Brucken@i3statprobe.com