Paper 081-2009

One-Step Change from Baseline Calculations

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 data set 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.

INTRODUCTION

The most common algorithm for computing change from baseline involves first physically dividing the original data set into baseline and post-baseline data sets. 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 data set containing the baseline value for every subject is created, and then merged with the post-baseline data set 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 data set 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 data set 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 data set, the second to process the baseline values separately, and the third to actually compute changes from baseline. Granted, the last two data sets combined form the original data set, 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 data set 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 data set 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 data set 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 data set. 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 data set 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 data set.

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.

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. http://www.lexjansen.com.

Chakravarthy, Venky, 2005. “RETAIN or NOT? Is LAG Far Behind?” PharmaSUG 2005 Conference Proceedings. 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