Paper 99-28

# The DOW (not that DOW!!!) and the LOCF in Clinical Trials

Venky Chakravarthy, Ann Arbor, MI

## ABSTRACT

Pharmaceutical companies conduct longitudinal studies on human subjects that often span several months. It is unrealistic to expect patients to keep every scheduled visit over such a long period of time. Despite every effort, patient data are not collected for some time points. Eventually, these become missing values in a SAS® data set later. For reporting purposes, the most recent previously available value is substituted for each missing visit. This is called the Last Observation Carried Forward (LOCF) and is familiar to the Pharmaceutical audience.

There are a number of ways to carry forward the last observation using SAS. This paper focuses on a technique that has gained prominence among SAS-L regulars. It is called the DO loop of Whitlock (DOW) also known as the Dorfman-Whitlock loop.

## INTRODUCTION

We mostly use the default behavior of the DATA STEP to create working code. However, certain common tasks are made easier by overriding the default behavior. In the Pharmaceutical Industry one such common task is LOCF. We will examine how to override one of the default behaviors of the data step to facilitate the calculation of LOCF.

To get value out of this a good understanding of Base SAS is assumed. If you are a beginner and work in the pharmaceutical or biotech industry, you are encouraged to read the paper and attend the presentation.

### THE ORGANIZATION OF THIS PAPER

(1) We will create a weight data set and also display what it should look like after LOCF.
(2) We will calculate the LOCF with the default behavior of the data step.
(3) Next we will examine how to modify this default behavior to benefit LOCF.
(4) We will finish with a few other benefits of such a modified data step behavior.

### CREATING A SAMPLE WEIGHT DATA SET

Since this paper is targeted for a Pharmaceutical audience, let us create a weight data set with a limited number of variables and observations. For the sake of simplicity, we will not deal with any treatment groups and consider only two patients with 5 scheduled visits for weight measures. We will also make the assumption that these two patients missed a few of their scheduled visits:

```
1.  data weight ;
2.    do pt = 1 to 2 ;
3.      do visit = 1 to 5 ;
4.        weight = ceil ( 200 * ranuni ( 1963 ) ) ;
5.        if weight < 150 then weight = . ;
6.        output ;
7.      end ;
8.    end ;
9.  run ;
```

+FSVIEW: WORK.WEIGHT (B)

| Obs | pt | visit | weight |
|---|---|---|---|
| 1 | 1 | 1 | . |
| 2 | 1 | 2 | . |
| 3 | 1 | 3 | 172 |
| 4 | 1 | 4 | . |
| 5 | 1 | 5 | . |
| 6 | 2 | 1 | . |
| 7 | 2 | 2 | 174 |
| 8 | 2 | 3 | . |
| 9 | 2 | 4 | 150 |
| 10 | 2 | 5 | . |

RANDOMIZATION

Visits 1 and 2 are Screening Visits. The 3rd visit is the randomization visit when patients are assigned to treatment groups. This is also the baseline visit. So, if a patient missed this visit, the weight from Visit 2 is considered the base weight. If the patient missed that too then weight at Visit 1 is considered the baseline. If that was also missing then the baseline weight is considered missing. Visit 4 is a post-baseline visit and 5 is the same but also the termination visit. If the termination weight was not recorded then the weight must be taken from visit 4 if available. If weight at visit 4 is missing then the termination weight is considered missing.

**HOW IT SHOULD LOOK AFTER THE LOCF**

Now that we have examined the data, let us take a look at the task we have, that of creating LOCF values. More particularly, let us examine how the projected BASELINE WEIGHT data set should look.

```
+FSVIEW:  WORK.BASEWEIGHT (B)-----+
| Obs    pt   visit  weight  BASE |
|                                 |
|  1     1       3     172   172  |
|  2     2       3       .   174  |
|                                 |
+---------------------------------+
```

The POST BASELINE data set should look as follows:

```
+FSVIEW:  WORK.POSTWEIGHT (B)-----+
| Obs    pt   visit  weight  POST |
|                                 |
|  1     1       4       .     .  |
|  2     1       5       .     .  |
|  3     2       4     150   150  |
|  4     2       5       .   150  |
|                                 |
+---------------------------------+
```

Let us now examine the code to go about this task. The traditional method of calculating LOCF for the baseline and post baseline data is first examined. This is what we have earlier referred to as the default behavior of the data step.

## LOCF - THE TRADITIONAL WAY

The following code will look remarkably familiar if you are working in the pharmaceutical industry.

```
1. data baseweight ;
2.     set weight ( where = ( visit <=3 ) ) ;
3.     by pt ;
4.     retain base ;
5.     if first.pt then base= . ;
6.     if weight ^= . then base = weight ;
7.     if last.pt ;
8. run ;
```

This is straightforward, simple and easy to follow. However, there are many explicit instructions.

(1) We RETAIN BASE to prevent its value from being reset to missing at the beginning of the next iteration of the data step.
(2) We also take care not to carry the previous patient's weight by reinitializing the BASE value to missing in line #5.
(3) We instruct to output only the last qualified baseline visit.

The calculation of LOCF for the post baseline visits is not much different. The only exceptions are:

(1) We subset from visit 4.
(2) We output all observations after (1).

The code for the POST BASELINE dataset is as follows:

```
1. data postweight ;
2.     set weight ( where = ( visit > 3 ) ) ;
3.     by pt ;
4.     retain post ;
5.     if first.pt then post = . ;
6.     if weight ^= . then post = weight ;
7. run ;
```

**THE INTERNAL WORKINGS OF THE DATA STEP**

Let us examine the internal workings of the data step that creates BASEWEIGHT. There are two compile time directives. The RETAIN statement and the BY PT group that follows the SET statement. We will deal with RETAIN here.

The directive RETAIN causes the variable that is created by an input or assignment statement to retain its value from **one iteration of the data step to the next** (SAS OnlineDoc, Version 8, 2001).

By iteration it is meant that the data step begins a loop from the top and ends it at the bottom by default. The bottom is typically a RUN statement or any step boundary like another DATA statement or a PROC. Under normal circumstances, as used in the creation of BASEWEIGHT here, the data step ***reads an observation from the WEIGHT data step every time it iterates*** from the top to the bottom. Mark the above statement carefully for future reference. The current iteration number manifests itself in the automatic variable _N_.

We will now examine what happens as the data step reaches the bottom:

**AT THE BOTTOM**

These default actions are performed by the data step (SAS OnlineDoc, Version 8, 2001).

1. Automatically, writes the observation to the BASEWEIGHT data set.
2. Automatically returns to the top of the data step.

**WHEN IT RETURNS TO THE TOP**

When it **returns to the top**, it automatically resets the Program Data Vector (PDV) to missing for the non-retained variables. Note that the variables in the input data set WEIGHT, are automatically retained but are immediately clobbered by the values from the next observation. Since the next iteration has begun, the iteration counter _N_ is incremented by 1.

We have covered **two very important concepts** as they relate to the default behavior of the data step. One **occurs at the top** and the other **at the bottom** as outlined above. We will next examine an alternate approach to the LOCF calculations and outputting only the last visit per patient. It will become clearer why the concepts outlined thus far have such an important bearing.

**MODIFYING THE DEFAULT BEHAVIOR – AN ALTERNATE APPROACH TO LOCF**

Let us now rewrite the code that creates the BASEWEIGHT data set as:

```
1.  data baseweight ;
2.     do until ( last.pt ) ;
3.        set weight ( where = ( visit <=3 ) ) ;
4.        by pt ;
5.        if weight ^= . then base = weight ;
6.     end ;
7.  run ;
```

Do not be deceived by the fact that the overall code has reduced by only 1 line. There are some significant improvements to the design. We have eliminated the RETAIN, the REINITIALIZING of BASE value for the next patient and do not explicitly ask the last value for the patient to be output (IF LAST.PT). That is 3 significant eliminations for such a simple example.

How is this accomplished? **Recall that in the default behavior of the data step, the non-retained variables are automatically reset to missing at the top of the data step**. So to eliminate the need for a RETAIN, the first task is to modify the default behavior so that the data step does not reset the values for BASE at the top. At least, we do not want it to reset the value until all the observations are read for a patient. So we would like the data step to return to the top only after reading all the visits for a patient. This task is accomplished by the DO loop of Whitlock (DOW) which puts the SET statement inside a DO UNTIL (LAST.PT) loop.

Ian Whitlock posted the DOW on SAS-L sometime in 1999-2000. At least, that was the first time it was prominently noticed. Paul Dorfman immediately recognized its immense potential and started applying it to various problems from many industries that represent SAS-L members.

So, how is the DOW used to eliminate the RETAIN in our data step. This is accomplished by putting the SET statement inside a DO loop with an UNTIL clause to end at the last qualified visit for the patient (LAST.PT). In this case it ends at Visit 3, since the input data set WEIGHT is subset to read only the first 3 visits.

The data step reads all the visits per patient inside the DOW without reaching the bottom of the data step. This means the data step does not iterate until all the visits are exhausted for a patient. Any variable created inside this do loop is not reset to missing until it exits the DOW. So the variable BASE is automatically retained for all the observations of any given patient.

Next we deal with the end of the loop. This special DO loop tests for the LAST.PT at the bottom of the DO loop. **Recall that by default, at the bottom of the data step, the current observation is written to BASEWEIGHT**. After exiting the DOW, only the last visit for the patient is available for output. Hence this is the only visit that is output for the patient. This is a very convenient feature since this eliminates the need to explicitly subset for LAST.PT.

Next we cover how the value of BASE is automatically cleaned up for the next patient. The data step returns to the top and begins its next iteration only after it exits the DOW and performs the default actions at the bottom. It exits the DOW only after reading all the observations for a patient. **Recall that in the default behavior of the data step, the non-retained variables are automatically reset to missing at the top of the data step**. Well, we have not retained BASE so it is automatically reset to missing at the top of the data step. It is as simple as that.

Likewise the POSTWEIGHT data set can be coded with the DOW as:

```
1.  data postweight ;
2.     do until ( last.pt ) ;
3.        set weight ( where = ( visit > 3 ) ) ;
4.        by pt ;
5.        if weight ^= . then post = weight ;
6.        output ;
7.     end ;
8.  run ;
```

Notice that the only difference from the DOW in BASEWEIGHT (other than the WHERE subset) is the explicit OUTPUT statement. This ensures that every observation inside the do loop is output and avoids the default output at the bottom of the data step. The only action performed at the bottom then is to automatically return to the top of the data step.

## CONCLUSION

We have seen an application of a technique that modifies the default behavior of the data step. The DOW has a wide variety of uses and is a natural selection when it comes to outputting a single observation per group and calculating variables that need to be retained within the group only. This is a common situation in Clinical Trials with LOCF.

## REFERENCES

SAS OnlineDoc (2001), *SAS Language Reference Concepts: Data STEP Execution*. SAS Institute Inc., Cary, NC, USA.

## ACKNOWLEDGMENTS

First and foremost my sincere appreciation goes to Ian Whitlock for authoring this technique. Without Paul Dorfman, this technique may have taken longer to come to light. So, my sincere thanks go to him for demonstrating some brilliant applications using this technique. Also, the many SAS-L participants who posted many interesting questions that made the DOW popular.

### CONTACT INFORMATION (HEADER 1)

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

Venky Chakravarthy
1591 Abigail Way
Ann Arbor, MI 48103
Email: swovcc@hotmail.com