Paper CT10

LOCF Programming with 6-step Index Mapping

Homer Wang, Covance Inc., King of Prussia, USA

ABSTRACT

Last Observation Carry Forward (LOCF) is a common method in efficacy analysis. It is error-prone and time consuming due to different methods and data situations. The 6-step index mapping method is introduced in this article which includes: 1. Create LOCF dataset with qualified LOCF feeding records; 2. Define desired time points; 3. Identify the missing time points by index; 4. Map in the missing time points by index; 5. Define the created LOCF data structure; 6. Define the keeping/dropping variables. This method has been implemented in multiple studies and proved to be accurate and time saving.

INTRODUCTION

There are few methods to create LOCF in analysis data program data step. One of the common methods is to use retain command. The challenges include but not limited to:

1. Select the qualified feeding records;
2. Populate the correct time points;
3. Decide which variables to LOCF;
4. Efficiently validate the LOCF dataset.

The 6-step index mapping mothed is introduced step by step in this article.

Step 1: Create LOCF dataset with qualified LOCF feeding records

In this step, the subset conditions per statistical analysis plan (SAP) are applied onto the original collected (OC) dataset with all other variables derived. The subset dataset contains only the qualified LOCF feeding records. Then only the last record of each time point is retained in the dataset.

```
**Create LOCF dataset with all the qualified LOCF feeding records;
data LOCF1t;
  set &indat;
      if (AVAL ne . and
          AVISITN in (&visit) and
          PARAMCD in (&test) and
          … all other conditions;
run;
```

```
**Select the last record for each time point;
proc sort data=locf1t; by USUBJID AVISITN ADTM; run;

data locf1t;
  set locf1t;
    by USUBJID AVISITN ADTM;
    if last. AVISITN;
    * Locfavis is the Index key to define which record need LOCF;
    locfavis = avisitn;
run;
```

Step 2: Define desired time points

The last dataset from step 1 is used to create a shell dataset which contains all the required time points for analysis.

```
** Create macro variables contain the analysis visit for each parameter;
%global vis_list;
proc sql noprint;
  select distinct avisitn into :vis_list separated by ', ' from locf1t
  where avisitn>0; /*vis_list contains a list of time points used in array
                     loop next*/

  select distinct compress(avisit, ' '), avisitn  into :vislist separated by
  ' ', :visnlist from locf1t where avisitn>0
  order by avisitn; /*visilist contains a list of time points in array
                     loop to fill in the missing value*/

  select distinct count(unique(paramcd)) into :paramct
  from locf1t where avisitn>0; /*paramct contains a list of parameters used
                                 in array loop next*/

  select distinct count(unique(avisit)) into :avisitct
  from locf1t where avisitn>0; /*avisitct is a counter for the array loop
                                next*/
quit;

** Create a shell dataset with all the parameters and time points;
data locfshels (drop=i j);
  length PARAMCD $8;
  set locfshels;
      by USUBJID;
  do j=1 to &paramct;
    PARAMCD = put(j, paracd.);
        do i=&vis_list;
          AVISITN = i;
          output;
        end;
  end;
run;
```

Step 3: Identify the missing time points by index
Transpose the last dataset from step 1 to identify all the missing time point.

```
proc sort data=locf1t; by USUBJID PARAMCD; run;

proc transpose data=locf1t out=tranlocf prefix=&prefix;
  by USUBJID PARAMCD;
  var locfavis; /* Index to indicate the time point which needs LOCF*/
  id AVISITN;
run;
```

Step 4: Map in the missing time points by index
The missing time points from the dataset created by step 3 are LOCF filled with an array data step. Then the dataset is transposed to create a LOCF dataset will non-missing time point.

```
 ** Fill in the missing index where the LOCF is from;
data tranlocff (drop=i);
  set tranlocf;
      array locfd [*] &vislist;
      do i = 1 to (&avisitct - 1);
        if locfd [i+1] eq . and  locfd [i] ne . then
            locfd [i+1] = locfd [i];
      end;
run;
```

The missing time points (yellow highlighted) in table created in step 3 are LOCF filled in (green highlighted).

USUBJID	PARAMCD	NAME	WEEK2	WEEK4	WEEK8	WEEK12	WEEK14	WEEK16	WEEK20	WEEK24
GS-US-417-0302-12345	PHGADA	LOCFAVIS	2	4	8	12	14	16	20	24
GS-US-417-0302-12354	PHGADA	LOCFAVIS	2	4	8	12		16	20	24
GS-US-417-0302-13245	PHGADA	LOCFAVIS	2	4	8		14			
GS-US-417-0302-13254	PHGADA	LOCFAVIS	2	4	8	12	14			
GS-US-417-0302-14235	PHGADA	LOCFAVIS	2	4	8	12	14	16	20	24

USUBJID	PARAMCD	NAME	WEEK2	WEEK4	WEEK8	WEEK12	WEEK14	WEEK16	WEEK20	WEEK24
GS-US-417-0302-12345	PHGADA	LOCFAVIS	2	4	8	12	14	16	20	24
GS-US-417-0302-12354	PHGADA	LOCFAVIS	2	4	8	12	12	16	20	24
GS-US-417-0302-13245	PHGADA	LOCFAVIS	2	4	8	8	14	14	14	14
GS-US-417-0302-13254	PHGADA	LOCFAVIS	2	4	8	12	14	14	14	14
GS-US-417-0302-14235	PHGADA	LOCFAVIS	2	4	8	12	14	16	20	24

```
** Transpose the LOCF filled dataset into the final LOCF dataset;
proc transpose data=tranlocff out=backlocf;
  by USUBJID PARAMCD;
run;
```

The final LOCF table with all the time point filled (green highlighted).

USUBJID	DTYPE	PARAMCD	AVISITN	LOCFAVIS	AVAL
GS-US-417-0302-12345	LOCF	PHGADA	12	8	52
GS-US-417-0302-12354	LOCF	PHGADA	16	14	57
GS-US-417-0302-13245	LOCF	PHGADA	20	14	57
GS-US-417-0302-13254	LOCF	PHGADA	24	14	57
GS-US-417-0302-14235	LOCF	PHGADA	16	14	50
GS-US-417-0302-14253	LOCF	PHGADA	20	14	50
GS-US-417-0302-12345	LOCF	PHGADA	24	14	50
GS-US-417-0302-12354	LOCF	PHGADA	12	8	52

Step 5: Define the created LOCF data structure
There are two possible dataset structures with LOCF: The dataset contains two completely separated OC and LOCF sets and the dataset with LOCF fill only the missing time points. The latter option is taken in this article.

```
**Dataset with all analysis time points including the LOCF filled time points;
data locfall1;
  merge locfshels (in=aa) backlocff (in=bb);
  by USUBJID PARAMCD AVISITN;
  if aa and bb;
run;
```

Step 6: Define the keeping/dropping variables
Keep or drop the variables from the dataset created in step 5.

```
**Dataset with all analysis time points including the LOCF filled time points;
data locfall1;
  merge locfshels (in=aa) backlocff (in=bb);
  by USUBJID PARAMCD AVISITN;
  if aa and bb;
  keep &keepvar;
  *drop &dropvar;
run;
```

CONCLUSION

The 6-step index mapping LOCF method resolved the issues encountered with other methods such as the method with retain command. Combine the 6 step with in one macro will improve the reusability and can be efficiently used at global study level.

REFERENCES

(1) Last Observation Carried Forward Versus Mixed Models in the Analysis of Psychiatric Clinical Trials, Robert M. Hamer, et al, Am J Psychiatry 166:639-641, June 2009.
(2) LOCF Method and Application in Clinical Data Analysis, Huijuan Xu, NESUG 2009 Posters

(3) SAS Support: A Macro for Last Observation Carried Forward. Jonson Jiang
https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2692-2018.pdf

CONTACT INFORMATION

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

Homer Wang
Principal Analysis Programmer
Covance Inc.
Email: Homer.Wang@Covance.com