P07 - 2008

Blinding Sponsors for Open Label Studies: Challenges and Solutions

Quan Jenny Zhou, Shengyan Hong, and Zhengping Ma
Eli Lilly and Company, Indianapolis, IN

ABSTRACT

Although double blinding (blind treating physicians and patients) is the optimal approach to minimize the bias in clinical research, it's not always feasible to conduct double blind studies. For open label studies, it's often desirable to blind the study sponsors to reduce potential bias and increase credibility of trial results. However, open label studies usually create some challenges to blind sponsors. In this paper, we discuss some types of Case Report Form (CRF) data that could unblind sponsors and then propose two methods to mask the data in order to blind sponsors. We share some SAS macros to implement the proposed methods and also provide a real example for illustration.

INTRODUCTION

Double blinding has been the preferred approach when investigating a new drug or regimen in clinical research, particularly for placebo-controlled trials. However, it is not always feasible to conduct double blind studies. For instance, in cancer studies, the investigational drug and the control drug often have different dosing schedules (e.g., bi-weekly schedule vs. 21-day schedule) and/or different routes (e.g., IV vs. oral), which makes blinding practically difficult or impossible. As a result, open label studies are often conducted for practical reasons.

Even for open label studies, it's still desirable blind the study sponsor to reduce potential bias due to the sponsor knowing the treatment level aggregated data while the study is ongoing. The practice helps increase credibility of study results and thus should be followed, particularly for registration studies.

In open label studies, in addition to treatment assignment data itself, many other data elements in the CRF data can reveal treatment assignment. Those data elements include (but not limited to):

- A common variable that is collected for all treatment arms but takes different values for different treatment arms. Take the dosing and treatment schedule for example, for Drug A it may be 500 mg daily dosing while for Drug B it could be 100 mg/m^2 every 6 weeks.
- A common variable that is collected for all treatment arms but the data collection frequency is different for different treatment arms. Continue the example in the above bullet point, a patient on Drug B can only have at most 1 dose adjustment per visit (every 6 weeks) while a patient on Drug A can have dose adjustment as often as daily.
- A variable that is only collected for a specific treatment arm. For example, Drug B is administered at the clinic or hospital by the health care professionals so treatment compliance is not a concern and thus not collected while Drug A is oral tablets and taken at home by patients so treatment compliance data is collected.

In order to blind the study sponsor, all the above mentioned data elements, among others, need to be blinded so that they can no longer give away the treatment assignment information. On the other hand, the blinded data should be appropriate to be used for generating statistical programs. That is, blinding should preserve important relationships in the data. In this paper, we propose two methods to blind those specific data elements but we believe that the methods can be generalized and applied to other data elements that need to be kept blinded to the study sponsor in an open label study.

METHODS

We propose two methods to blind the data elements that can reveal the treatment assignment information in an open label study. After blinding the data, the sponsor personnel (such as physicians and statisticians) can access the blinded data. Meanwhile, since it is not practical to restrict everyone only to the blinded data, the personnel who can access the unblinded data should be kept at minimum, and independent of study team whenever possible.

Depending on the needs, the data elements we discussed in INTRODUCTIOIN can be blinded by using the either or both of 1) randomly re-ordering the patient numbers/IDs for all the patients in the study; 2) replacing the values of a variable with one or more specific values.

We developed three SAS macros to implement the proposed blinding methods:

1. **%reorder_create**: for a specific variable from a dataset, this macro generates a re-ordered list of unique values for the variable.
2. **%reorder_apply**: this macro replaces the true values of a variable with the re-ordered values generated by %reorder_create.
3. **%imputval**: this macro replaces the values of a variable with a specific value or values.

The SAS code for these three macros is attached in the Appendix of this paper. We will illustrate the use of these SAS macros with an example.

EXAMPLE

Our example study is to compare Drug A and Drug B. A is supposed to be taken 500 mg daily, and B is 100 mg/m^2 once every 6 months. Since it is an open label study, the CRF pages are designed without any consideration of blinding, so there are many data elements that can give away the treatment assignment information. Our goal is to mask all these data elements so that the database won't reveal which treatment a patient is on. We first re-order the patient numbers/IDs in those datasets that need to be masked. To do this, we follow the next two steps:

Step 1: Call %REORDER_CREATE to generate a master dataset (UNIPAT) with a list of unique values for the original and re-ordered patient numbers/IDs. Note that the &inset parameter should be the dataset that contains all the available patient numbers/ID in the study. For our purpose, we used the VISIT dataset.

```
%reorder_create(inset     = inlib.visit,
```

```
                outset    = unipat,
                scrvar    = patient);
```

Step 2: Call %REORDER_APPLY to apply the re-ordered patient numbers/IDs to the specified dataset. This step should be repeated for all the datasets that need to be masked. For example, below is the SAS code to mask the treatment assignment dataset.

```
%reorder_apply(orderset  = unipat,
               inscrset  = inlib.trtasgn,
               outscrset = outlib.trtasgn,
               scrvar    = patient,
               relset    = inlib.trtasgn,
               relvar    = invid);
```

The result is displayed as follows:

Before

PATIENT	VISIT	DOSETYPE	DOSE
1	1	1 (Assigned)	500
2	1	1	100
3	1	1	500
4	1	1	100

After

PATIENT	VISIT	DOSETYPE	DOSE
2	1	1	500
4	1	1	100
1	1	1	500
3	1	1	100

Before re-ordering the patient number/ID, we could tell by the value of DOSE that patients 1 and 3 are on Drug A while patients 2 and 4 are on B. After re-ordering, we cannot tell whether patient 2 is truly on treatment A or not.

Similarly, this can blind those datasets that are only collected for one treatment arm. Take the treatment compliance dataset for example; as it is only collected for arm A, the compliance dataset would only include patients in arm A. After re-ordering the patient numbers, the (re-ordered) patient numbers in the compliance datasets are from both arms.

After re-ordering patient numbers, there could still be data elements that can identify treatment assignment information. For example, in the treatment administration dataset that collects the dosing information, Drug A may have multiple records for dosing adjustment per visit while B can only have one record per visit at most (see the below dataset for illustration). In this dataset, after re-ordering the patient numbers, we don't know whether patient 1 is truly in arm A, but we can still tell that some patient in arm A had a few dose adjustments. Because reduction or omission (DOSEADJ = 2 OR 4) are all caused by adverse events, we will be able to have some idea about treatment A's safety profile by aggregating dose reductions or omissions.

PATIENT	VISIT	DOSEADJ	DOSE	REASADJ
1	1	1 (increased)	550	2 (Protocol)
1	1	2 (reduced)	500	1 (AE)
1	2	3 (no change)	500	2
1	2	4 (Omitted)	0	1
1	2	1	500	2
2	1	2	80	1
2	2	3	100	2
2	3	3	100	2

In clinical trials data, it is quite often that two variables are logically dependent. In our example, reason for dose adjustment (REASADJ) is dependent on the type of adjustment (DOSEADJ). When the adjustment is a reduction or omission (DOSEADJ = 2 or 4), the reason can only be adverse event (REASADJ = 1); for the other types (DOSEADJ = 1 or 3), the reason will be per protocol (REASADJ = 2).

To further blind the data while keeping the logic relationship between DOSEADJ and REASADJ, we can use the below SAS code:

```
%reorder_create(inset     = inlib.trtadj,
                outset    = unidadj,
                scrvar    = doseadj);

%reorder_apply(orderset  = unidadj,
               inscrset  = inlib.trtadj,
               outscrset = outlib.trtadj,
               scrvar    = doseadj,
               relset    = inlib.trtadj,
               relvar    = reasadj);
```

Similarly, the value of DOSE could reveal the same information as dose adjustment (DOSEADJ). Because the value of DOSE is continuous, it's easier to replace the DOSE value with a certain number than re-ordering. Although the replaced value of DOSE might not logically match the re-ordered values of dose adjustment (DOSEADJ), it doesn't affect the analysis programming. To assign a specific value to DOSE, we can run the following SAS code:

```
%imputval(impset  = outlib.trtadj,
          impvar  = dose,
          impval  = 300,
          impmiss = n,
          outset  = outlib.trtadj);
```

And now the masked treatment adjustment data fully blind the treatment assignment information.

After

PATIENT	VISIT	DOSEADJ	DOSE	REASADJ
1	1	3	300	2
1	1	1	300	2
1	2	4	300	1
1	2	2	300	1
1	2	4	300	1
2	1	1	300	2

2	2	4	300	1
2	3	4	300	1

SUMMARY

We proposed two methods to blind different types of data elements that could be used to identify the treatment assignment. These methods are specifically useful to blind the study sponsor to the treatment level aggregated data for an open label study, thus helping reduce bias and improve credibility of the study results. We implemented the proposed methods through three SAS macros and demonstrated how to use them with an example.

We believe that the methods can be generalized and applied to other situations than those discussed in this paper. However, we also recognize that there could be scenarios that may need a new or different method.

Finally, we'd like to point out that probably no algorithms or methods can prevent a determined mind from "breaking the code", particularly for open-label studies. It would be helpful to plan carefully when designing CRF. Efforts should be made to optimize the data collection in a way that can effectively blind sponsor after some data scrambling algorithms (such as the one discussed in this paper) are applied to the data.

Acknowledgement

The authors would like to thank Ilya Lipkovich for reviewing the paper and providing insightful comments.

Contact Information

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

Quan (Jenny) Zhou
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: zhouqu@lilly.com

Shengyan (Sam) Hong
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: hongsh@lilly.com

Zhengping Ma
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: mazh@lilly.com

Appendix: SAS Macros

Macro 1: %reorder_create

```
%macro reorder_create(inset     = ,
                      outset    = ,
                      scrvar    =);
%*******************************************************************;
%*     m a c r o   p a r a m e t e r   d e f i n i t i o n s       *;
%*******************************************************************;
%* inset      : the input dataset                                  *;
%* outset     : the output dataset                                 *;
%* scrvar     : the variable(s) in inset to be reordered           *;
%*******************************************************************;

  %*****************************************************************;
  %* get the unique non-missing values of the variable scrvar      *;
  %*****************************************************************;
  proc sort data = &inset out = _univar(keep = &scrvar) nodupkey;
    by &scrvar;
  run;

  %*****************************************************************;
  %* generate a random number using different seed every time      *;
  %* it is called.                                                 *;
  %*****************************************************************;
  data  _univar1;
    set _univar;
    randnum = ranuni(0);
    order   = _n_;
  run;

  %*****************************************************************;
  %* sort the data by the random number                            *;
  %*****************************************************************;
  proc sort data = _univar1;
    by randnum;
  run;

  data  _ordvar;
    set _univar1;
    order = _n_;
    %*-- create the new reordered variable -;
    rename &scrvar = p_&scrvar;
  run;

  %*****************************************************************;
  %* merge the original and reordered datasets together by the     *;
  %* order number                                                  *;
  %*****************************************************************;
  proc sort data = _univar1 out = _univarf;
    by order;
  run;

  proc sort data = _ordvar out = _ordvarf;
    by order;
```

```
  run;

  data work.&outset(drop = order randnum);
    merge _univarf
          _ordvarf;
    by order;
  run;

  %*******************************************************************;
  %* delete all the temporary datasets from work library             *;
  %*******************************************************************;
  proc datasets library=work;
    delete _univar _univar1 _univar2 _univar3 _ordvar _ordvar1;
  quit;

%mend reorder_create;
```

Macro 2: %reorder_apply

```
%macro reorder_apply(orderset  = ,
                     inscrset  = ,
                     outscrset = ,
                     scrvar    = ,
                     relset    = ,
                     relvar    = );
%*******************************************************************;
%*     m a c r o   p a r a m e t e r   d e f i n i t i o n s       *;
%*******************************************************************;
%* orderset : the dataset generated from the reorder_create macro  *;
%* inscrset : the input dataset with a variable to be reordered    *;
%* outscrset: the output dataset with the variable reordered       *;
%* scrvar   : the variable that to be reordered                    *;
%* relset   : the input dataset with variable(s) related to the    *;
%*            reordered variable. If left blank, no extra related  *;
%*            variable is needed in the output.                    *;
%* relvar   : the variable(s) that is related to the reordered     *;
%*            variable that need to be kept in the same order/pair *;
%*            with the reordered variable.                         *;
%*******************************************************************;

  %*******************************************************************;
  %* merge dataset to be reordered with dataset generated from       *;
  %* reorder_create macro containing the reordered data.             *;
  %*******************************************************************;

  proc sort data = &orderset out = _order;
    by &scrvar;
  run;

  proc sort data = &inscrset out = _scrset;
    by &scrvar;
  run;

  data _scrset2(drop  = p_&scrvar);
    merge _order (in = ina)
```

```
          _scrset(in = inb);
    by &scrvar;
    if inb;
    &scrvar = p_&scrvar;
  run;

  %if "&relset" = "" %then %do;
    data &outscrset;
         set _scrset2;
    run;
  %end;
  %else %do;
  %*********************************************************************;
  %* merge the reordered dataset with the related dataset (relset)   *;
  %* to get the related variables with the reordered variable.       *;
  %*********************************************************************;
    proc sort data = &relset
              out = _reldata(keep = &scrvar &relvar) nodupkey;
      by &scrvar;
    run;

    proc sort data = _scrset2(drop = &relvar);
      by &scrvar;
    run;

    data &outscrset;
      merge _scrset2(in = ina)
            _reldata(in = inb);
      by &scrvar;
      if ina;
    run;
  %end;

  %*********************************************************************;
  %* delete all the temporary datasets from work library             *;
  %*********************************************************************;
  proc datasets library=work;
    delete _order _scrset _scrset2 _reldata);
  quit;

%mend reorder_apply;
```

Macro 3: %imputval

```
%macro imputval(impset  = ,
                impvar  = ,
                impmiss = y,
                outset  = ,
                impval  = );
%*********************************************************************;
%*     m a c r o   p a r a m e t e r   d e f i n i t i o n s       *;
%*********************************************************************;
%* impset  : the input dataset                                     *;
%* impvar  : the variable in impset with values to be replaced     *;
%* impmiss : replace missing value or not? y=yes and n=no.         *;
%*           default : y                                           *;
```

```
%* outset  : the output dataset                                          *;
%* impval  : the pre-defined value to replace the current value with *;
%*********************************************************************;
  data &outset;
    set &impset;


    %if %upcase("&impmiss") = "Y" %then %do;
       &impvar = &impval;
    %end;
    %else %if %upcase("&impmiss") = "N" %then %do;
       if missing(&IMPVAR) = 0 then &impvar = &impval;
    %end;
  run;
%mend imputval;
```