Paper DH14

Handling Data "Blinding" for Oncology Open Label Studies Using a SAS Macro

James Zhao

Merck & Co. Inc., Upper Gwynedd, PA

INTRODUCTION

In Oncology, many clinical trials are open label studies due to the need to treat life threatening or late stage cancers. Both investigators and subjects know what drug is being used and what treatment group a subject belongs to. However, once data is collected and transferred to a pharmaceutical sponsor, it is often necessary to blind the data in order to reduce the potential bias during data cleaning, analysis and reporting process. For data traceability, it is desirable to store the data in the clinical repository as it is collected but blinded to internal team during analysis. This raised a clear business need to blind the data after being extracted from a clinical trial database. A clean solution to meet this requirement is to develop a SAS® blinding macro and include it as part of the data extraction. This paper examines the design logic of this macro, the approach of inputting blinding requirements, the creation of a global macro variable to facilitate the looping through of the entire library, the method to re-order the blinded variables, and the verification of the blinding outputs.

GENERAL DESIGN OF THE MACRO AND THE PARAMETERS

A utility macro on0blinding() is developed to blind specified variables for Oncology open label studies. The macro will read in a SAS metadata dataset containing the variables to be blinded and mask all the variables listed in the metadata that exist in the data. The macro has 4 parameters allowing some flexibility for the end users.

```
%macro on0blinding(input_libname = lptsdd
      ,output_libname = lptss
      ,input_blinding_dataset = lonmtd.oncolblind0vars
      ,debug = N);
```

For the short term this macro is run by an unblinded programmer after the raw data are extracted from the clinical repository database. After the macro is executed the unblinded programmer will QC the blinded data to ensure that they are properly blinded, and then copy them to the working protocol folder so all blinded programmers and statisticians have access.

For the long term, the plan is for IT to execute this macro as part of the data extraction from the clinical repository. The macro will read the extracted data from the SAS work library and write the blinded data out to the SAS work library. Other IT utilities will then copy the blinded data to a working protocol folder.

A parameter input_blinding_dataset is used to store a SAS dataset with variables to be blinded. The blinding requirements are collected in an excel spreadsheet for flexibility. The spreadsheet is then converted into a permanent SAS dataset and stored in a departmental standard metadata folder called lonmtd. Below is a portion of the spreadsheet for illustration purpose.

Spreadsheet containing variables to be blinded

SDTM Domain	SDTM Variable	Blinded Value
AE	ELEMENT	Blinded
AE	ETCD	Blinded

DM	ACTARMCD	Blinded
DM	ACTLARM	Blinded
DM	ARM	Blinded
DM	ARMCD	Blinded
DM	DYSONTRT	999
EX	EXDOSE	999
EX	EXROUTE	Blinded
EX	EXSPID	Blinded

Blinded Value of 999 indicates that the particular variable is a numeric type.

STEPS THAT IMPLEMENT THE BLINDING

Step 1: Read in input_blinding_dataset containing the variables to be blinded.

```
proc sort data=&input_blinding_dataset out=spec;
   by sdtm_domain sdtm_variable;
run;
```

In case there is a need to add or drop certain variables to be blinded per specific protocol, the end user can update the spreadsheet and re-convert it to SAS dataset and then pass into the macro.

Step 2: Create a variable containing all blinded variables per domain.

```
data spec_cont;
   length sdtm_variable_r $200;
   set spec;
   by sdtm_domain;
   retain sdtm_variable_r '';
   if first.sdtm_domain then sdtm_variable_r = sdtm_variable;
   else sdtm_variable_r = strip(sdtm_variable_r) || ' ' ||
      strip(sdtm_variable);
   if last.sdtm_domain then do;
      rename sdtm_variable_r = sdtm_variable;
      output;
   end;
   keep sdtm_domain sdtm_variable_r;
run;
```

Variable sdtm_variable contains the entire list of variables to be blinded for each domain and will be used later in the process to identify the variables to be blinded.

Step 3: Create a macro variable containing all blinded variables.

```
proc sql noprint;
   select distinct sdtm_variable
   into :allvars separated by ' '
   from spec
   order by sdtm_variable;
quit;
```

A global macro variable &allvars is created by using proc sql into: in order to facilitate the merge later.

Step 4: Transpose spec dataset to make all sdtm_variable in a single row for each domain. Since the spec dataset has a vertical structure it is necessary to transpose it to be a horizontal structure in order to merge it with the SDTM datasets by each domain.

```
proc transpose data=spec out=spec_1 prefix=spec_;
   by sdtm_domain;
   id sdtm_variable;
   var blinded_value;
run;

*** add sdtm_variable to each domain;

data spec_2;
   merge spec_1 (in=a) spec_cont;
   by sdtm_domain;
   if a;
run;

data spec_3;
   set spec_2;
   call symput(sdtm_domain,strip(sdtm_variable));
   rename sdtm_domain=domain;
run;
```

Step 5: Loop through the entire input library and mask variables that need to be blinded per the spec.

```
data tables(keep=memname);
   set sashelp.vtable(where=(libname=upcase("&input_libname")));
run;

proc sql noprint;
   select count(memname) into :tot
   from tables;
quit;

proc sql noprint;
   select memname into :tabname1 - :tabname&tot
   from tables;
quit;

%do _j=1 %to &tot;

data &&tabname&_j;
   merge &&input_libname..&&tabname&_j (in=a) spec_3
(where=(domain=upcase("&&tabname&_j")));
   by domain;
   if a;
```

We will then parse the global macro variable &allvars, retrieve each value separated by space, and then replace the value of each variable with the blinded value from the spec, achieving the desired blinding purpose.

```
 %let _i = 1;
   %do %while(%length(%scan(&allvars,&_i,%str( ))));
       %local _var&_i;
       %let _var&_i=%scan(&allvars,&_i,%str( ));
       %put _var&_i=&&_var&_i;

      if spec_&&_var&_i ne '' then &&_var&_i = spec_&&_var&_i;
```

```
      %let _i=%eval(&_i+1);
   %end;
run;
```

Step 6: Re-order per the original variable order for each domain and drop those temporary variables created during the process.

```
proc contents data = &&input_libname..&&tabname&_j
   out = vars(keep = varnum name) noprint;
run;

proc sql noprint nowarn;
   select distinct name
   into :orderedvars separated by ' '
   from vars
   order by varnum;
quit;


data &&output_libname..&&tabname&_j ;
   retain &orderedvars;
   set &&tabname&_j;
   keep &orderedvars; *** Only keep variables that are in the original domain;
run;

%end;
```

Step 7: Verify the blinding outputs. It is the end user's responsibility to perform a QC on the output datasets to ensure that proper blinding is done per specification. This QC can be completed by proc freq for each domain and by performing some manual checking of each datasets to be blinded.

CONCLUSION

Macro on0blinding() is developed, validated and executed in Oncology open label studies. It successfully masks variables that could potentially reveal subject's treatment group information.

REFERENCES

AUTHOR CONTACT INFORMATION

James Zhao
Principle Scientist, Statistical Programming
Merck Research Laboratories
UG1CD-38
PO Box 1000
North Wales, PA 19454
Phone: 267-305-7672
Email: yi_zhao@merck.com