

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

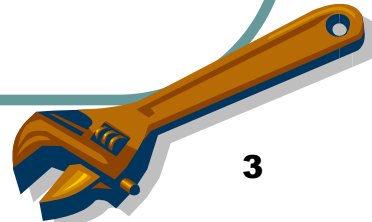
James Zhao
Merck & Co., Inc.

Topics Covered

- Introduction
- General design of the macro and the parameters
- Steps that implement the blinding
- Conclusion

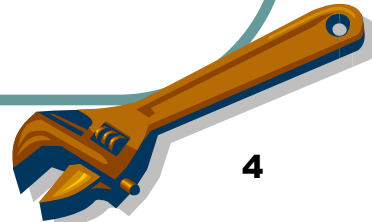
Introduction

- In Oncology, many trials are open label due to the need to treat life threatening or late stage cancers
- Once data is collected and transferred to a sponsor, it is often necessary to blind the data 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



Introduction (cont.)

- It 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 and development of this blinding macro



General Design of the Macro and the Parameters

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

General Design Of The Macro and the Parameters (Cont.)

Short term implementation:

- Unblinded programmer extract unblinded data
- Run this macro
- Verify blinding effects
- Copy outputs to blinded protocol folder

General Design Of The Macro and the Parameters (Cont.)

Long term implementation:

- IT execute this macro as part of the data extraction
- The macro reads the extracted data from the SAS work library and writes the blinded data out to the SAS work library
- Other IT utilities then copy the blinded data to a working protocol folder

General Design Of The Macro and the Parameters (Cont.)

- The macro reads in a SAS metadata dataset containing the variables to be blinded
- 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

General Design Of The Macro and the Parameters (Cont.)

- The spreadsheet is then converted into a permanent SAS dataset and stored in a departmental standard metadata folder

General Design Of The Macro and the Parameters (Cont.)

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

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;
```

Steps That Implement The Blinding

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

Steps That Implement The Blinding

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

Steps That Implement The Blinding

```
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;
```

Steps That Implement The Blinding

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

Steps That Implement The Blinding

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

Steps That Implement The Blinding

Step 4: Transpose spec dataset to make all sdtm_variable in a single row for each domain.

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

Steps That Implement The Blinding

```
*** 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;
```

Steps That Implement The Blinding

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;
```

Steps That Implement The Blinding

```
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;
```

Steps That Implement The Blinding

We will then parse the global macro variable &allvars, retrieve each value separated by a 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;
```

Steps That Implement The Blinding

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;
```

Steps That Implement The Blinding

```
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;
```

Steps That Implement The Blinding

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.

Contact Information

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