PharmaSUG 2018 – Paper SI-15

One Project, Two Teams: The Unblind Leading the Blind

Part 2: Options

Kristen Reece Harrington, Rho, Inc.

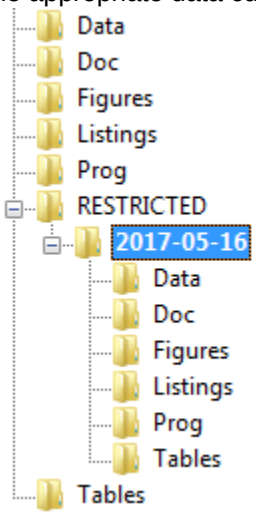
ABSTRACT

In the pharmaceutical world, there are instances where two independent programming teams exist to ensure blinded treatments and data are maintained by the appropriate parties. For the purposes of this discussion, blinded corresponds to scrambled data and unblinded corresponds to actual data. Within these projects, blinded programmers use scrambled data to create programs which produce both blinded and unblinded outputs. To ensure the blind is maintained and ethics are upheld, only the blinded programming team produces and modifies the programs. While robust programming is key, there are multiple techniques that can be used to ensure that the blinded programs remain unedited during the unblinded runs. We will discuss the pros and cons of two techniques: utilizing %include statements and making files read-only programmatically.

INTRODUCTION AND GENERAL ASSUMPTIONS

This discussion focuses on studies requiring both blinded and unblinded deliverables produced by the same vendor. Typically, this dual team approach is required for safety review meetings (ex. DSMB) where both blinded and unblinded reports are delivered during the course of the study. To facilitate this discussion, let's agree on the following assumptions:

- The work supports a DSMB
- GRID environment
- Programs are on one network rather than multiple servers
    - The Main directory (Unrestricted) will contain blinded (scrambled) data
    - The RESTRICTED directory will contain unblinded data
        - IT restricts access to the RESTRICTED directory
    - The RESTRICTED directory is nested within the Main directory. Both directories have same structure; however, the RESTRICTED directory contains an additional dated folder to ensure the appropriate data cut is being utilized



- There are two teams
    - Blinded Team hence forth referred to as Team B
        - Responsible for all programming activities
        - Working with blinded treatments and possibly scrambled data if there is an unblinding risk associated with specific data points, visit schedules, etc.

    - Unblinded Team hence forth referred to as Team U
        - Responsible for running programs created by Team B on actual data with actual treatments
        - Will not modify or create any programs. Any required program modifications to support validation will be communicated to Team B by Team U
- Unblinding occurs at the SDTM level
- UAT data are available for initial programming and study setup

%INCLUDE STATEMENTS

Team B is responsible for all programming activities, including creation of the setup program and switch macros. The setup program consists of all appropriate libnames, SAS® options for automatic inclusions and all appropriate output paths. All paths within the setup program should contain macro variables specified within the Switch macro. The Switch macro should be a very simple macro program containing a minimum of two macro variables. The first macro variable specifies the type of data used (blinded or unblinded). The second specifies the paths. Additionally, these macro variables can be used to efficiently macrotize the treatment columns within displays.

Basic good programming practice dictates that macro programs should contain well documented header blocks. This is extremely true for the Switch macro. Furthermore, additional documentation within the Switch macro is strongly encouraged.

Example of a Switch macro:

```
/*-------------------------------------------------------------------*

   PROJECT:    Specify your study

   MACRO:      Switch.sas

   PURPOSE:    SWITCH BETWEEN DSMB OPEN AND CLOSED DISPLAYS

   ARGUMENTS:  1st => <FAKE, DSMB> FAKE=scrambled codes,DSMB=real codes>
               2nd => <Date of restricted directory for DSMB YYYY-MM-DD>

   RETURNS:    GLOBAL MACRO VARIABLES:
                  SwitchDir => DSBM Subdirectory (e.g. \RESTRICTED\<DATE>)
                               Includes leading \ for easy substitution
                               into path specification
            %*Switch(FAKE,); *Goes to Main study folders;
            %*Switch(DSMB,2017-05-17); *Goes to Restricted folders;

   USAGE:      Called by %Setup_TLF

   PROGRAM HISTORY:

   DATE        PROGRAMMER        DESCRIPTION
   ---------   ---------------   ----------------------------------------
   01May2017   KHarrington       created
*-------------------------------------------------------------------*/

%macro switch(BLIND,Date_);
    %GLOBAL Switch SwitchDIR TrtLbl1 TrtLbl2 ;
    %let SWITCH = %upcase(&BLIND);
    /*EXCECUTE THIS CODE WITH DUMMY DATA*/
    %if %upcase(&BLIND) = FAKE %then %do;
```

```
            %let SwitchDir = \RESTRICTED\&Date_;
            %let TrtLbl1 = Group A;
            %let TrtLbl2 = Group B;
      %end;
      /*EXECUTE THIS CODE WITH REAL TREATMENTS*/
     %if %upcase(&BLIND) = DSMB %then %do;
            %let SwitchDir = ;
            %let TrtLbl1 = Control;
            %let TrtLbl2 = Actual Treatment Label;
      %end;
  %mend switch;
  %*Switch(FAKE,); /*Goes to Main Study Folder*/
  %Switch(DSMB,2017-05-16); /*Goes to Restricted Directory*/
```

The Switch macro is included using an %include statement within the Setup macro program. The &SwitchDir macro variable created in the Switch macro should be utilized for all path specifications, such that the appropriate libnames are utilized and all files are written out to the appropriate locations. All SDTM, ADaM, Table, Listing, and Figure (TLF) programs will include the Setup macro by way of an %include statement.

A utility macro is used to create the programs for use by Team U. The utility macro has two parameters: Blinded Study path (&StudyPath) and Unblinded Study path (&NewPath). The macro scans the directories which contain SDTM, ADaM, Table, Listing, Figure and all subsequent validation programs. A list of program names is generated which is then used to create macro variables for name, type and iterative number for linking. A do loop is utilized to parse through the macro variables, and output a new SAS program with the appropriate name in the appropriate unblinded location (i.e. SDTM programs are written to the unblinded SDTM folder). The following is used to populate the file with only an %include statement.

```
 %do i=1 %to &dnum;
    %let dsn = %scan(&dsns,&i," "); %*Name of file;
    %let type = %scan(&dts,&i); %*Type of file - table, listing, figure
original location;
    %let indir = &STUDYPATH.\&type;
    %let full = &STUDYPATH.\&type.\&dsn.;
    %put dsn=&dsn   type =&type   fullpath=&full;


   %*-------------------------------------------------------------------;
   %* Copy programs, remove everything, and add only ;
   %* an include statement to original program               ;
   %*-------------------------------------------------------------------;


     DATA _null_;
       file   "&NewPath.\&type\&dsn";
       put  "%include '&full' / source2;";
     RUN;
  %end;
```

The above code is located within the utility macro, inside a do loop that parses the macro variables using their iterative number assignment. Note the last four lines of the code above is where the actual file containing only an %include statement is created. The utility macro can also be modified to address only one file or one specific file type (i.e. SDTM) rather than all files.

PROS VS CONS - %INCLUDE STATEMENTS

There are several pros and cons associated the use of %include statements.

PROS:

1. As the programs in the Restricted directory (Team U) are one line of code, any modifications to the restricted programs are easily noted.
2. The programs in the Restricted directory are always current as the %include statement will always pull the most recent file.

CONS:

1. Due to the use of the Switch macro, Team B and Team U cannot run programs concurrently.
2. The same %Setup_TLF file is used by both teams.
3. Partially masked data signifies the treatments headers are masked as a single letter (A, B, C, etc) and treatment mapping is unknown to the reviewers. This can be accomplished using formats; however, with the use of %include statements, the actual data must reflect the masked values. If a safety signal is noted, the data would need to be regenerated to determine the actual treatment assignment for a specific data point rather than being able to use the data generated to create the masked datasets and displays.
4. The logs of the %include files can be difficult to read. It may be necessary to view the original full program to determine where warnings or errors may be occurring.

While the number of cons seem to outweigh pros, this is still an effective method.

OPTION 2: CHANGING ATTRIBUTES

The premise of the second option we will be discussing is the same as noted above. We still the same directory structure and two teams, Team U and Team B. For this option, however, we utilize two different %Setup macros and do not have a %Switch macro. The processing of path assignments driven by the %Switch macro in Option 1 is contained within the %Setup macro in Option 2.

A similar utility macro is used to generate programs for Team U with slight differences. The same process to create macro variables associated with each file type and file name is used. However, rather than outputting a file containing only an %include statement, the full file is output to the Restricted directory and the attribute of the file is changed to Read-Only. Further, our utility macro has three parameters, rather than two: Team B's Unrestricted Study Path (&StudyPath), Team U's Restricted Study Path (&NewPath), and First Run (&FirstRun). The &FirstRun parameter indicates whether this is the first time the utility macro is being called or not. If &FirstRun=Y, there are no files in the Restricted area and the macro proceeds to copy programs and change the attribute to Read-Only. If &FirstRun=N, files do exist in the Restricted area. The attributes are changed to Read-Write. The files are deleted, and then replaced with Read-Only versions of the updated files from the Unrestricted area.

```
%*-------------------------------------------------------------------------------------------------------;
%* Copy programs, update the call to setup_tlf and make the files read only       ;
%*-------------------------------------------------------------------------------------------------------;
```

```
 %do i=1 %to &dnum;
   %let dsn = %scan(&dsns,&i," "); %*Name of file;
   %let type = %scan(&dts,&i); %*Type of file - table, listing, figure original location;
   %let indir = &STUDYPATH.\&type;
   %let full = &STUDYPATH.\&type.\&dsn.;
   %let newfull = &NewPath.\&type.\&dsn.;
   %put dsn=&dsn   type =&type   fullpath=&full;

    DATA _null_;
     infile "&indir.\&dsn";
     file   "&NewPath.\&type.\&dsn.";
     input;
         _infile_ = tranwrd(_infile_,"&StudyPath.","&NewPath.");
       put _infile_;
    RUN;

    /*MAKE READ ONLY*/
    %let rc=%sysfunc(system(attrib +r "&newfull"));
 %end;
```

Note that the last %let statement is where the files attributes are changed to Read-Only. The utility macro can also be modified to address only one file or one specific file type (i.e. SDTM) rather than all files.

PROS VS CONS – CHANGING ATTRIBUTES

There are several pros and cons associated changing file attributes.

PROS:

1. Team B and Team U can run programs concurrently.
2. Modifications to programs are very traceable due to the date/time stamp changing if such modification occurred, and the process of changing the attributes generating file specific metadata.
3. The logs are very easy to read and reference along with the full program.
4. Partial masking of treatments can be handled using formats in the %Setup file such that the data is accurate and easy to reference if a safety signal is noted.

CONS:

1. Any modifications by Team B require Team U to rerun the utility macro to update the programs on the Restricted side.
2. Communication is key. Team B must notify Team U when a program has been modified or a new program has been created in order to keep the programs on the Restricted side current.

CONCLUSION

While the author recognizes there may be multiple techniques utilized to maintain the blind when two teams are in place, we focus on the two noted within this paper. Based on the pros and cons outlined above, the process of changing attributes is more robust. The changing attributes process saves valuable time by allowing both teams to run programs concurrently. The logs are easily read which allows notes, warnings and errors to be quickly and concisely communicated by Team U to Team B for program modifications. Partial masking can be handled using formats which allows for appropriate and accurate communications with a safety review board to discuss potential safety signals, without having to modify the data. The cons noted with the process become very limited with effective communication.

CONTACT INFORMATION

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

Kristen Reece Harrington
Rho, Inc.
919-595-6377
Kristen_Harrington@rhoworld.com