Paper 050-29

Comparing the Desired Variable Definitions with the Actual Data Sets Across Multiple Studies

Shane L. Hornibrook, PRA International, Charlottesville, VA

ABSTRACT

The flexibility of the SAS® System offers a variety of methods by which programmers can compare data across studies to ensure that the datasets and variables meet quality assurance standards. In industry where quality control (QC) is paramount, analysts must choose data comparison methods which provide the most reliable comparisons. Precise programming of datasets to meet FDA and client approval is of the utmost importance in the pharmaceutical industry. To meet that challenge programmers must perform QC on many different levels. When provided with data definition documents, a programmer must be able to quickly and efficiently compare the specifications to the datasets across multiple studies, and prepare a methodology for categorizing and scoring the differences between specifications and datasets. This is especially paramount for Contract Research Organizations (CROs) where programming efforts for multiple studies with the same client are running concurrently. Deviations from specifications must be adequately documented and adjusted as needed.

INTRODUCTION

In this paper we will be covering a hypothetical client with five studies. Variable definitions change on an irregular basis, and updates are sent in Microsoft Word format. The CRO is required to track and adjust variables in ongoing studies. The variable definition data is extracted from Microsoft Word using DDE, reducing the potential for human error. The differing variable definitions are tabulated using SASHELP.VCOLUMNS. The output of the variables needing adjustment is enhanced by the use of the ODS system, with colorful markup indicating the aspect(s) of the variable that is need of adjustment (type, length, format, and/or label), as well as the definition from the variable definition document. Scoring factors are applied to the differences using the SAS function SPEDIS (spelling distance). Issues can therefore be prioritized, and handed off to the appropriate analyst for adjustment or clarification with the client.

GRABBING THE DATA FROM WORD

Using DATA _NULL_ statements and DDE we can automate the extraction of the data from the Microsoft Word documents. We will have SAS start Word, create bookmarks, and iterate through the document. An alternative option would be saving the Microsoft Word document into a format that preserves the breaks between cells and rows (html, excel, or even a text file) and iterating through the file to create the dataset. This option is less than ideal as it introduces an extra step to the process, and extra steps increase the amount of QC work required.

CREATING A DDE CHANNEL TO WORD

The following method for starting and manipulating Word via a DDE channel were extensively covered by William W.Viergever and Koen Vyverman (2003);

```
OPTIONS noxsync noxwait xmin;
FILENAME wordsys DDE 'winword|system' LRECL=5000;

 DATA _NULL_;
    LENGTH fid rc start stop time 8;
    fid=fopen('wordsys','s');
    IF (fid le 0) THEN DO;
       rc=SYSTEM('start winword');
       start=DATETIME();
       stop=start+10;
       DO WHILE (fid le 0);
         fid=FOPEN('wordsys','s');
         time= DATETIME ();
         IF (time GE stop) THEN fid=1;
       END;
    END;
    rc=FCLOSE(fid);
RUN;
```

OPENING AND PRE-PROCESSING THE WORD FILE

The Microsoft Word tables have extraneous line feeds and carriage returns. These line feeds will make the task of importing the data more difficult, so we will send DDE commands to Word to find and replace them with backslashes. When viewing the imported table, it is obvious that the backslashes are not part of the information in the cells. For QC purposes, making the original line breaks obvious in the newly-created SAS dataset is preferred to replacing the breaks with spaces or other non-printing characters.

```
DATA _NULL_;
   FILE wordsys;
   PUT '[AppMinimize]';
   PUT '[ChDefaultDir "\\path\to\specifications\",0]';
   PUT '[FileOpen.Name="VariableDefinitions"]';
   PUT '[EditReplace.Find="^p",.Replace="\\\",
                        .Direction=0,.ReplaceAll,.Wrap=0]';
   PUT '[EditReplace.Find="^l",.Replace="\\\",
                        .Direction=0,.ReplaceAll,.Wrap=0]';
RUN;
```

ADDING BOOKMARKS, IF NECESSARY

Next we iterate through the document, adding a bookmark for each table. In this example we will bookmark the first twenty tables. If the document already has the tables bookmarked, this step is unnecessary, and the tables may be extracted merely by specifying the bookmark names. In this manner we can extract multiple tables, or only one.

```
%MACRO add_bookmarks;
OPTIONS mprint symbolgen source source2;

/* make sure we are at the top of the document */
DATA  _NULL_;
   FILE wordsys;
   PUT '[StartOfDocument(0)]';
RUN;

%DO j = 1 %to 20; /* There are 20 tables of interest in our Word document */

   DATA _NULL_;
     FILE wordsys;
     /* Move to the next table and select the table */
     PUT '[EditGoto.Destination="t"]';
     PUT '[TableSelectTable]';
     /* Add a bookmark, incorporating the iteration number */
     PUT "[EditBookmark.Name=%str(%")table&j.%str(%")",.Add]";
   RUN;

   /** Make a DDE triplet based on the filename and bookmark */
   FILENAME wordtab&j. DDE "winword|VariableDefinitions!table&j." NOTAB;

%END;

/* return to the top of the document */
DATA  _NULL_;
   FILE wordsys;
   PUT '[StartOfDocument(0)]';
RUN;

%MEND add_bookmarks;

%add_bookmarks;
```

Each iteration of loop “j” sends DDE commands to move to the next table and add a bookmark. The filename statement adds a DDE triplet for input into a SAS dataset.

PULLING DATA INTO SAS AND CREATING INDIVIDUAL SAS DATASETS

Next we will pull each table into SAS and save as a SAS dataset. Be sure to leave enough room for the contents of each variable. Certain variables have only a few records containing a great deal of information, while the majority of the table contains nearly empty variables. As these are long character variables we are able to use `OPTIONS compress = yes` to reduce the number of bytes required to represent the data.

```
%MACRO input_tables;
OPTIONS compress = yes;
%DO j = 1 %TO 20; /* We are interested in the first twenty tables */
```

```
   /* Read  first cell of the table to determine the dataset name, in the format c_dataset */
   DATA _NULL_;
      LENGTH temp $255. tablename $8.;
      /* Temp: the text of the first line is a description of the table
         Tablename: Tablenames are only 8 characters long */
      INFILE wordtab&j. DSD MISSOVER DLM='09'x OBS=1;
      /* we are only interested in the first cell of the table.*/
      INPUT temp;
      temp = REVERSE(LEFT(temp)); /*reverse the string*/
      /* read the characters between the end bracket of the temp
         variable up to and including its prefix: c_ */
      tablename = SUBSTR(temp, INDEX(temp,')')+1, INDEX(temp,'_c'));
      /* reverse the tablename variable */
      tablename = COMPRESS(REVERSE(tablename));
      /* put the tablename variable into a macro variable */
      CALL SYMPUT ('tablename', tablename);
   RUN;

 /* Now that we have the dataset name we read in the dataset again. Data starts at line 4. */
   DATA &tablename.;
      LENGTH dataset $8 variable $8 label $100 type  $30
             possible_values $1000  format $100 comments $2000;
      INFILE wordtab&j DSD MISSOVER DLM='09'x FIRSTOBS = 4;
      INPUT variable label type possible_values format comments;
            dataset = "&tablename.";
   RUN;
   %END;

   %MEND input_tables;

   %input_tables;
```

CROSS TABULATING THE VARIABLES

We have now pulled a set of tables from a MS-Word variable definition document into individual SAS datasets. We could combine the datasets into a single SAS dataset, but our purposes we will leave them separate. It is easy to browse and compare these spec datasets to the MS-Word document. Our next step is to compare the variables in our datasets with each other and to our variable definition document. As per the methods used by Rucker(2003) SASHELP.VCOLUMN provides us an easy location for all our metadata in any specified libraries.

First we assign unique names to the libraries containing the datasets we are going to compare. Here we are using short acronyms for each study to try to minimize the width of our future report.

```
   LIBNAME s1 '\\path\to\study01\';
   /* Libnames s1 through s4 */
   LIBNAME s5 '\\path\to\study05\';
```

SASHELP.VCOLUMN is a SAS view which contains metadata for all datasets within a library. Rather than using PROC CONTENTS we can use the metadata in SASHELP.VCOLUMN to extract information about our variables. We use SASHELP.VCOLUMN to pull only the information from libraries containing client data.

```
   DATA vcolumns(keep= libname memname name type length type_temp label format x);
      LENGTH type $8.;
      SET SASHELP.VCOLUMN(rename=(type=type_temp));
      IF upcase(libname) in ('S1' 'S2' 'S3' 'S4' 'S5'); /* only libraries of interest */
```

We concatenate the type and length into a single variable in order to compare it to the incoming variable definition documents.

```
      type =  COMPRESS(type_temp||length);
      x = 'X'; /* X will be used as a symbol in our report */
   RUN;
```

This creates a dataset that contains one line per library/dataset/variable combination. We want to compare across studies, so we need to transpose the dataset.

TRANSPOSING THE DATASET TO MAKE A 'WIDE' REPORT

Since we are comparing across studies, it is easier to read the differences in a vertical report (wide) rather than how the variables come out of SASHELP.VCOLUMN. PROC TRANSPOSE provides a simple method for 'flipping' the report;

```
PROC SORT data = vcolumns;
   BY memname name type label format ;
RUN;

PROC TRANSPOSE data=vcolumns out=wide (drop=_name_);
   ID libname;
   VAR x;
   BY memname name type label format ;
RUN;
```

We are then able to see the presence and absence of variables by scanning for 'holes' in our report. In this contrived example, programmers in study one (S1) and study two2 (S2) have coded Age as a numeric variable (8 byte) whereas programmers in studies three (S3) through five (S5) have coded it as a character variable with a length of three.

memname	name	type	Label	format	S1	S2	S3	S4	S5
C_KEYVAR	AGE	Num8	Age	3	X	X			
C_KEYVAR	AGE	Char3	Age				X	X	X

FLAGGING MORE THAN ONE DEFINITION

In order to highlight variables with more than one definition we will use a PROC SQL left join and a distinct count of the number of memname, name combinations. Then we add a variable containing the number of observations where there is more than one definition.

```
PROC SQL;
  CREATE TABLE wide AS
  SELECT a.*, n_defn
  FROM wide AS a left join
     (SELECT DISTINCT memname, name, count(name) AS n_defn
      FROM wide group BY memname, name ) AS b
     ON a.name eq b.name AND a.memname eq b.memname AND n_defn > 1
     ORDER BY a.memname, a.name;
  QUIT;
RUN;
```

JOINING THE ACTUAL VARIABLES WITH THEIR DEFINITIONS

Since we are trying to ensure that the analysis goes smoothly, we will attach the variable definitions to our cross tabulation. We have already extracted the variable definitions using DDE and massaged the DDE'd dataset. In order to catch any variables in the specifications that may not be in the database yet, we will use a full join.

```
PROC SQL;
   CREATE TABLE wide AS
   SELECT a.label AS spec_label, upcase(b.type) AS spec_type, a.possible_values,
          a.format as spec_format, a.comments, b.*
   FROM C_KEYVAR AS a FULL JOIN wide AS b
   ON upcase(a.dataset) EQ upcase(b.memname) AND upcase(a.variable) EQ upcase(b.name)
   WHERE memname IN ('C_KEYVAR')
   ORDER BY memname, name, type, label, format, s1, s2, s3, s4, s5;
   QUIT;
RUN;
```

ADDING THE CONTRACT RESEARCH ORGANIZATION'S VARIABLE DEFINITION DOCUMENT

Just as the client holds variable specifications in a Microsoft Word document, so too does the Contract Research Organization (CRO). The CROs document has been updated with modifications to the variable definitions and additions to the specifications as per email and telephone conversations with the client. It is expected that that any deviations from a client specification document would be explained by notations in the CRO's modified specification document. In the same manner as described above we can add variables such as cro_label, cro_type, cro_format, with the ultimate aim of reconciling differences between the CRO's version of the specifications (modified through dialogue with the client), and the clients version of the specifications.

FLAGGING THE VARIABLES WITH OUT-OF-SPEC DEFINITIONS

In order to ease the job of the analyst responsible for rectifying the differences between the variables in different datasets we want to highlight differences within the document. To quantify and prioritize the variable adjustments we will apply a scoring algorithm: SPEDIS, spelling distance. SPEDIS quantifies the difference between character variables. A simplified scoring of the actual format, type, and label vs. the client specifications format, type, and label is presented below.

```
DATA temp1;
   SET temp1;
   type_score = SPEDIS(label,spec_label);
   format_score = SPEDIS (label,spec_label);
   label_score = SPEDIS (label,spec_label);
RUN;
```

After the generation of the scoring algorithm, the analyst may choose to concentrate first on amending the discrepant variable types, then move on to the formats and labels.

OUTPUTTING THE REPORT WITH ODS

Finally we want to output a nicely marked-up report. Using the Output Delivery System (ODS) we have the option to create a document in RTF, HTML, or PDF format. The user can choose a format suitable to the report or audience. In this case we are dealing with a rather “wide” report, so Microsoft Excel is the destination application. SAS can create an HTML document with the extension .XLS -- suitable for opening in Microsoft Excel. We will be using PROC REPORT to generate the body of the report. We will be computing the colors for the individual cells using a compute block and a style.

```
ODS HTML FILE = "\\path\to\report\directory\Variable_comparison_&sysdate..xls";

PROC REPORT DATA = wide nowd CENTER STYLE(REPORT)={background=white}
            STYLE(HEADER)={foreground=blue font_weight=bold font_size=1}
            STYLE(COLUMN)={background=white foreground=black font_size=-1};

COLUMN ("A comparison of PRA Datasets with Specification Documents" ("Program run on
        &sysdate &systime by &sysuserid" n_defn  type_score format_score label_score
       ("PRA dataset properties" memname name type format label)
       ("Present in Client studies" S1 S2 S3 S4 S5)
       ('Type' cro_type spec_type) ('Label' cro_label spec_label)
       ('Format' cro_format spec_format) ('Comments' comments_cro comments) ));
 DEFINE n_defn       / display ' ' noprint;
 DEFINE type_score   / display ' ' noprint;
 DEFINE label_score  / display ' ' noprint;
 DEFINE format_score / display ' ' noprint;
 DEFINE memname      / 'Datasets'      ;
 /* other define statements have been left off for brevity **/
```

Compute blocks can be used to change the attributes of the cells in the report. In this case we want to highlight the name of the variable where there is more than one definition.

```
 COMPUTE n_defn;
   IF (n_defn > 1) THEN
   CALL DEFINE('_C6_', "Style", "STYLE=[BACKGROUND = red  font_weight=bold]");
 ENDCOMP;
```

If highlighting cells is not sufficient we can also add a commentary using compute blocks. In this compute block we want to highlight those cells that have any problem with their type/length.

```
 COMPUTE comments_cro /character LENGTH=8;
   /* Any number of compute blocks can be added using the same layout as the following */
   IF type_score > 0 THEN DO;
      IF spec_type = ' ' THEN comments_cro = "Type not in client spec;";
      ELSE comments_cro = "Type, Length differs from client spec;";
      IF cro_type = ' ' THEN comments_cro = trim(comments_cro)||" Type not in our spec; ";
      CALL DEFINE(_COL_,"STYLE","style(CALLDEF)={background=orange foreground=orange}");
      CALL DEFINE('_C7_',"STYLE","style(CALLDEF)={background=red font_weight=bold}");
   END;

 ENDCOMP;

RUN;

ODS HTML CLOSE;
```

VIEWING THE FINAL REPORT

The final report, an HTML document with an .XLS extension can be easily opened and viewed by the analyst. To appraise the progress of the variable adjustment task, the analysis, or lead programmer can re-run the report on a regular basis.

The final report lends itself well to online or landscape viewing. A compressed version is included below;

A comparison of PRA Datasets with Specification Documents																	
Program run on 01DEC03 9:27 by HornibrookShane																	
PRA dataset properties					Present in Client studies					Type		Label		Format		Comments	
Datasets	Variable	Length, Type	Format	Label	S1	S2	S3	S4	S5	CRO	Client	CRO	Client	CRO	Client	CRO	Client
C_KEYVAR	AGE	NUM8	3	Age (years)	X	X				NUM8	NUM8	Age (years)	Age (years)	3	3		Years between birth and enrollment date
C_KEYVAR	AGE	CHAR3		Age (years)			X	X	X	NUM8	NUM8	Age (years)	Age (years)	3	3	Type/ Length differs from client spec; Format differs from client spec;	Years between birth and enrollment date

CONCLUSION

Ongoing meta-QC of a programming effort can be made foolproof by using The SAS System. SAS has been shown to be useful for pulling data directly from specification documents, using DDE, and combining it with live snapshots of datasets under revision. This method provides the ability to quickly pinpoint issues within datasets such as misnamed variables and even misspellings in variable labels. Prior to the utilization of this report to QC SAS programming efforts an exhaustive (and exhausting) QC of PROC CONTENTS listings was required. The tools provided by SAS provide quick and robust methods for meta-reporting of programming efforts across studies and address issues as they arise.

REFERENCES

Rucker, D. ""Haven't We Met Somewhere Before?" and Other Useful Lines (of code) to Get to Know Their Data" *Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, paper 241-27, 2002*

Pass, R., Macneil, S., "Proc Report: Doin' it in Style" *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference, paper 15-28, 2003.*

Viergever, W.W., and Vyverman, K., "Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange -- Against All ODS, Part II" *Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference, paper 16-28, 2003.*

ACKNOWLEDGMENTS

Thanks to SAS-L for letting me lurk for the last few years. Thanks to Tim Williams, and the rest of my co-workers at PRA International.

CONTACT INFORMATION

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

Shane Hornibrook
PRA International
4105 Lewis and Clarke Drive
Charlottesville, VA, 22911
(434) 951-3433
HornibrookShane@PRAIntl.com or SUGI29@ShaneHornibrook.com
http://www.prainternational.com