PharmaSUG 2017 - Paper DA06

Check Your Data: Tools for Automating Data Assessment

Paul Stutzman, Axio Research LLC

ABSTRACT

This paper presents tools that can be employed to check the consistency and validity of data, and it discusses building blocks for automating these tasks. Individually or in combination, these components can be used to check data within data sets, across data sets, and over time.

It is important to carefully examine data to ensure consistency and accuracy. This can be a time-consuming process, so automating these tasks can be of great value. These data checking tasks range from simple (e.g. making sure related variable pairs like --TEST/--TESTCD are consistent within a data set) to more complex (e.g. seeing if VISIT/VISITNUM pairs are consistent across all data sets, or looking at how the structure of a set of data sets and the information they contain change over time). Regardless of the complexity, there are some fundamental building blocks that can help automate these processes.

INTRODUCTION

This paper starts with some brief examples of tools that can be useful when assessing data. These examples include tools for assessing data within and across a set of data sets, tools that assess how data and data structures change over time, and tools that examine the timestamps of related objects. It also discusses ways automation can improve these tools.

The main focus of the paper is on creating these tools. It discusses SAS®-based components for getting information about data sets, variables, and the information they contain. It shows techniques for combining and presenting information from these components in meaningfully ways.

Within the context of this paper, “automation” refers to programmatically determining data sets, variables, and/or data of interest and performing analyses based on these findings. One example would be programmatically finding all data sets that contain visit and visit number variable pairs, and seeing if the values in these pairs are consistent within and across a set of data sets. Implementation of the Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM) standards can greatly simplify these automation tasks, since they enforce consistency in the attributes and content of data sets and variables.

TOOLS FOR ASSESSING DATA

This section briefly describes some tools that can be used to assess data. It also discusses how automating these tools can be beneficial. Subsequent sections describe components that can used to build these tools, and how these example tools were built.

The goal is not to provide a comprehensive list of tools. Rather, it is to show some possibilities and to inspire other creative uses of the building blocks that follow.

ASSESSING RELATED VARIABLES WITHIN DATA SETS

Some of the most straight-forward and easiest data assessment tools to implement are programs that check the content of related variables within individual data sets to ensure consistency. A few examples include VISIT/VISITNUM, --TEST/--TESTCD, and --TEST/--STRESU variable pairs from SDTM compliant domains; QNAM/QLABEL variable pairs in SDTM compliant supplemental qualifier domains; and PARAM/PARAMCD/PARAMN variables in ADaM compliant analysis data sets.

As mentioned, these programs are easy to write for individual data sets or domains. However, a great deal of power and efficiency can be gained when these programs are automated – when they programmatically identify which data sets or domains contain the variables of interest, when they automatically perform checks for consistency, and when they present the results in a comprehensive, concise and effective manner.

ASSESSING RELATED VARIABLES ACROSS DATA SETS

A slightly more complex set of assessment tools involve looking at related variables across multiple data sets or domains. One example is checking all VISIT/VISITNUM variable pairs to ensure consistency across all SDTM domains within a study. Discrepancies can be highlighted in the output in order to make them more obvious. Sample output from a program that does this is shown as Output 1.

Compare VISIT/VISITNUM Pairs Across All Domains

VISIT	EG	LB	MB	MS	PE	SV	TV	VS	XO
SCREENING	1	1	1	1	1	1	1	1	
DAY 1			2	2			2		
DAY 7		8	8	8	8	8	8	**9**	
DAY 13		14	14	14		14	14	14	
DAY 14			**14.01**	**Missing**					
EOS		18	18	18	18	18	18	18	18

Output 1: Output from a VISIT/VISITNUM checking program

The automation component programmatically determines which domains have VISIT/VISITNUM variable pairs, it finds the all the values contained in those pairs, it checks for consistency, and it presents the results effectively.

ASSESSING RELATED VARIABLES ACROSS TIME

A variant of checking related variables across multiple data sets or domains is checking related variables across different versions of the same data set or domain over time. This can be invaluable for ensuring consistency between different data transfers or snapshots. Output 2 shows an excerpt from output that compares values of the related variables QLABEL and QNAM from SDTM supplemental qualifier domains, with their corresponding domains from a previous data snapshot.

Compare QLABEL/QNAM Pairs

For the SUPPCM Domain

	QLABEL	Old QNAM	New QNAM
New QLABEL	Conmed Treament for Adverse Event 3		AELINE3
QLABEL Not in New	Tconmed reament for Adverse Event 3	AELINE3	
	Concomitant Medication Route Others	ROUTEOTH	
QNAMs Match	Conmed Treament for Adverse Event 1	AELINE1	AELINE1
	Conmed Treament for Adverse Event 2	AELINE2	AELINE2

Output 2. Output from a QLABEL/QNAM checking program

Within each domain, it identifies value pairs that only appear in one of the snapshots, and those that appear in both. The automation piece consists of programmatically determining which supplemental qualifier domains appear in both transfers, performing the comparisons, and presenting the results of the comparisons effectively.

ASSESSING ALL DATA SETS AND VARIABLES OVER TIME

Assessing structural and data content changes that happen between data transfers and snapshots can be extremely important. It can be critical to know if data sets or variables have been added or omitted, or if variable attributes have changed. Evaluating how the number of observations in each data set change is also important. Often it is necessary to know if new or different values are present in categorical variables, or if numeric variables' ranges have changed. These are just some of the assessments that can be done when evaluating sets of data over time.

This is a lot of information, so one effective way of presenting it is in individual spreadsheets within a Microsoft EXCEL workbook. Particulars regarding the specific information in these spreadsheets was presented in a previous PharmaSUG paper (Stutzman, 2016). Automation pieces include programmatically determining and comparing the data sets and variables in each transfer, identifying and comparing the values contained in categorical and numeric variables, and presenting the results.

ASSESSING CHRONOLIGICAL RELATIONSHIPS

It can be important to assess the chronological relationship between programs, their output, and their logs. Information about test or quality control (QC) programs, their output, and their logs can be added, providing a snapshot of the chronology of programming, output generation and QC. Example output is shown in Output 3.

Project X

	Main Object						Testing/QC							
	Exists?			Dates and Times				Exists?			Dates and Times			
Name	Object	Program	Log	Object	Program	Log	Name	Program	Log	Output	Program	Log	Output	Issues
ADAE	Yes	Yes	Yes	28JAN17:09:54	02DEC16:09:16	28JAN17:09:54	TADAE	Yes	Yes	Yes	06DEC16:12:32	19JAN17:10:27	19JAN17:10:27	
ADEX	Yes	Yes	Yes	28JAN17:09:54	02DEC16:14:47	28JAN17:09:54	TADEX	Yes	Yes	Yes	09DEC16:15:27	19JAN17:10:27	19JAN17:10:27	
ADLB	Yes	Yes	Yes	28JAN17:09:55	06DEC16:11:52	28JAN17:09:55	TADLB	Yes	Yes	Yes	06DEC16:13:41	19JAN17:10:27	19JAN17:10:27	
ADMB	Yes	Yes	Yes	02FEB17:08:46	23JAN17:10:50	02FEB17:08:46	TADMB	Yes	Yes	Yes	18JAN17:10:54	19JAN17:10:27	19JAN17:10:27	Test program is older than program. Program is newer than the test program log. Program is newer than the test output.
ADMH	Yes	Yes	Yes	28JAN17:09:54	02DEC16:08:34	28JAN17:09:54	TADMH	Yes	Yes	Yes	05DEC16:16:30	19JAN17:10:27	19JAN17:10:27	

Output 3. Output from a chronology checking program

Automation is accomplished by simply pointing the tool at a set of folders, and allowing standard folder structures and file naming conventions to drive the object and date comparisons.

COMPONENTS FOR BUILDING AND AUTOMATING ASSESSMENT TOOLS

This section describes components that can be used to create assessment tools like the ones described above. All these components are SAS-based. In a few instances (e.g. pipes) the specifics relate to SAS running in a Windows environment, but the concepts apply to any operating system.

STRUCTURAL INFORMATION

Information related to the structure of SAS data sets and their variables is important for many types of assessments. In addition to providing information that can be assessed directly, it is also a key to automation. Specifically, it enables programs to determine the data sets and variables to be assessed.

Structural information comes from three main sources: DICTIONARY tables, the CONTENTS procedure, and file system information gathered through pipes.

DICTIONARY Tables

SAS provides read-only access to a number of tables called DICTIONARY tables. They contain a wealth of information about the SAS environment and SAS data sets. Some of the most pertinent information for data assessment comes from the COLUMNS DICTIONARY table, which describes the all the variables in all data sets in the currently assigned LIBNAMEs.

Unlike normal SAS data sets, DICTIONARY tables can only be read by the SQL procedure. The following PROC SQL statements read the COLUMNS DICTIONARY table:

```
create table columns as
select *
from dictionary.columns
where libname ne "WORK";
```

The following fields in the COLUMNS table are of particular interest: LIBNAME, MEMNAME (data set name), NAME (variable name), TYPE, LENGTH, and LABEL. This information is useful for making comparisons between data sets and data transfers/snapshots. It can also be used to automate assessments by determining, for example, which data sets contain specific variables or sets of variables.

The CONTENTS Procedure

PROC CONTENTS can provide similar information to the COLUMNS DICTIONARY table by using the OUT= option and the _ALL_ keyword on the DATA= option as follows:

```
proc contents data=Old._all_ noprint out=OldData;
run;
```

In this example, all data set and variable information from the data sets in the library Old will be written to a data set named OldData.

In addition to the information available in the COLUMNS DICTIONARY table, the PROC CONTENTS output also contains information about the data sets themselves: the number of observations they contain, their creation and last modification dates, etc. Unlike DICTIONARY tables, the resulting data set can be read by DATA steps and regular SAS procedures – not just PROC SQL.

As with DICTIONARY table information, this information is useful for making comparisons between data sets and data transfers/snapshots, and it can also be used to automate assessments.

Pipes

Pipes provide the ability to submit commands to the operating system, and they make the output from those commands available via FILENAME statements. The following FILENAME statement will submit a Windows DIR command to read the C:\SASData folder and all its subfolders (by using the "/s" option):

```
filename DirInfo pipe "dir C:\SASData /s" lrecl=32767;
```

Output from the DIR command, just as it would appear in a Windows COMMAND window, will be available from the DirInfo FILENAME. It can be read as follows:

```
data Dir;
     length InRec $256;
     infile DirInfo length=LRecl;
     input InRec $varying256. LRecl;
```

Once this data has been parsed, the operating system's information about all files in this folder (and its subfolders) is available. Last modification dates and file sizes are of particular interest. This information is useful when assessing information about SAS program output, logs, and the programs themselves.

INFORMATION ABOUT DATA

In addition to obtaining information about the structure of data sets and the variables they contain, it is equally important to assess the values within these data sets. For the examples in this paper, this information comes primarily from the SUMMARY or MEANS procedures or from the FREQ procedure.

ODS OUTPUT and ODS TRACE

In many cases, the information needed from a SAS procedure can be obtained by generating output from the procedure directly, often with an OUT= option. However, some procedures can provide much more information than just what is contained in their OUT= output data sets. ODS OUTPUT and ODS TRACE provide the means for accessing this information programmatically.

ODS TRACE writes the names of the pieces of information available from SAS procedures to the SAS log. The following example shows how to find the names of the information that is available when the NLEVELS option is included in a PROC FREQ step:

```
ods trace on;
proc freq data=DataSet1 nlevels;
     tables Var1;
run;
ods trace off;
```

The resulting output that is written to the SAS log is shown in Output 4:

```
Output Added:
-------------
Name:         NLevels
Template:     Base.Freq.NLevels
Path:         Freq.NLevels
-------------

Output Added:
-------------
Name:         OneWayFreqs
Label:        One-Way Frequencies
Template:     Base.Freq.OneWayFreqs
Path:         Freq.Table1.OneWayFreqs
-------------
```

Output 4. SAS log output from an ODS TRACE of PROC FREQ's NLEVELS option

Once the name of the output is known (in this case it is NLevels) the ODS TRACE statements can be removed and an ODS OUTPUT statement can be added before the PROC of interest, as in the following:

```
ods output NLevels=Levels;
proc freq data=DataSet1 nlevels;
     tables Var1;
run;
```

After this code is run, NLevels information will reside in a data set named Levels.

Note: ODS OUTPUT will not produce an output data set when a SAS procedure's NOPRINT option is used. NOPRINT can be quite useful for suppressing printed (textual) output from procedure steps within automated data assessment programs, since these procedures might be called many times, and their printed output is of no use. A work-around is to use ODS EXCLUDE statements before and after SAS procedure steps as follows:

```
ods exclude all;
proc freq ...;
     ...
run;
ods exclude none;
```

ODS EXCLUDE ALL prevents the creation of textual output. This is similar to how adding the NOPRINT option to the PROC FREQ statement would have worked.

The SUMMARY and MEANS Procedures

Both PROC SUMMARY and PROC MEANS produce simple descriptive statistics, and their output is available for assessment via the OUT= option of an OUTPUT statement. The following example creates a data set with simple descriptive statistics for all numeric variables in DataSet1:

```
proc means data=DataSet1 noprint;
     var _NUMERIC_;
     output out=Means;
run;
```

In this case, the output data set (Means) could be compared with the output from a different data set to see if same named variables have similar values and ranges.

The FREQ Procedure

PROC FREQ is the primary tool for assessing related variables. It can also be used to identify categorical variables and to assess their values.

The OUT= option of the TABLES statement can be used to gather the information for related variables. The following example gets frequencies and percentages for all matched values of VISITNUM and VISIT in the LB SDTM domain:

```
proc freq data=SDTM.LB noprint;
     tables VISITNUM * VISIT / out=LBVis;
run;
```

The resulting output will be written to LBVis. This data set could be checked for consistency within the LB domain itself, it could be compared to visit information from previous data snapshots, or it could be checked for consistency with other domains.

PROC FREQ can create two particularly useful output data sets that are only available via ODS OUTPUT: OneWayFreqs and NLevels. OneWayFreqs provides one way frequency information for all variables that are included on a TABLES statement. The NLevels data set contains information about the number of unique values that variables contain.

PROC FREQ's OneWayFreqs Output

Given the following PROC FREQ TABLES statement:

```
tables Var1 Var2 Var3 / out=Freqs;
```

The resulting output data set (Freqs) will only contain frequency information for the last variable specified (Var3). The OneWayFreqs output data set provides a way around this. Placing the following statement before a PROC FREQ step will cause one-way frequency information to be written to AllFreqs for all variables in the previous TABLES statement

```
ods output OneWayFreqs=AllFreqs;
```

Be sure to heed the note regarding NOPRINT in the ODS OUTPUT and ODS TRACE section of this paper.

PROC FREQ's NLEVELS Option and NLevels Output

NLEVELS determines the number of unique values one or more variables contain. For example, if all the values of variable SEX are either "F", "M", or " ", the NLEVELS output would show three levels (two non-missing levels and one missing level). The following PROC FREQ step would accomplish this:

```
ods output NLevels=Levels;
proc freq data=Analysis.ADSL nlevels;
     tables SEX;
run;
```

The resulting output data set (Levels) is shown as Output 5.

	TableVar	TableVarLabel	NLevels	NMissLevels	NNonMissLevels
1	SEX	Sex	3	1	2

Output 5. NLevels output data set (Levels) for SEX

Comparing the number of levels to a threshold value can be useful for programmatically identifying variables that are likely to be categorical. The values of these variables can be tracked over time to see if categorical values are added, removed, or changed.

Be sure to heed the note regarding NOPRINT in the ODS OUTPUT and ODS TRACE section of this paper.

OUTPUT AND FORMATTING

There are a number of ways to effectively present the results of data assessment tools. ODS output in RTF, PDF, or HTML format can each be effective in many circumstances. The idea is to present the resulting information in a clear and concise manner in which discrepancies and issues are easily spotted.

Two tools that can assist are traffic lighting and making use of the ODS EXCEL XP tagset.

Traffic Lighting

"Traffic Lighting" within the context of this paper is the use of colors, bolding, etc. to highlight items of interest in order to make them easily identifiable. Automated tools, like the ones presented in this paper, can produce many pages of output. It is important that items of concern or interest jump off the page. Items of concern might include differences in VISIT/VISITNUM pairs across SDTM domains, variables that are missing in one group of data sets, or instances where a program's log predates the program itself.

One way to highlight a value in the REPORT procedure's output is to provide ODS formatting information along with a character value to be displayed. For example, if the current value in the variable Column2 is to appear bolded and in red, the following statement could accomplish this:

```
Column2 = cat("~{style [foreground=red fontweight=bold]", Column2, "}");
```

Note that PROTECTSPECIALCHARS=OFF must appear in the PROC REPORT DEFINE statements for variables that contain ODS formatting information, like the variable Column2 in the example above.

There are a number of other techniques for traffic lighting including the use of SAS FORMATS, PROC REPORT COMPUTE blocks, and CALL DEFINE. Some excellent papers have been written about traffic lighting in general (Hadden, 2006) and about traffic lighting within EXCEL workbooks (Black, 2011).

The ODS EXCEL XP Tagset

The ODS EXCEL XP tagset provides a powerful method for creating Microsoft EXCEL spreadsheets from within SAS. Automated tools can create large volumes of output, and spreadsheets within an EXCEL workbook can organize and present this information in an effective way. The specifics of using the ODS EXCEL XP tagset are beyond the scope of this paper, but some excellent papers have already been written on this topic (DelGobbo, 2014 and 2015).

AUTOMATION

The following building blocks are particularly helpful for automating assessment tools. Methods for passing values of interest via macro variables using CALL SYMPUT and PROC SQL's INTO keyword are discussed. CALL EXECUTE, which enables the execution of macros, DATA steps, and SAS procedures from within DATA steps is also described, as is the use of the _ALL_, _NUMERIC_, and _CHARACTER_ keywords.

Setting Macro Variables

Macro variable values can be set within macro code, but sometimes it is easier or more effective to do so in DATA steps or with PROC SQL steps. CALL SYMPUT performs this function in DATA steps. The INTO keyword of PROC SQL's SELECT statements provides similar functionality.

CALL SYMPUT

Sometimes the easiest way to determine the names of a group of data sets or variables of interest is with a DATA step. In these cases, using CALL SYMPUT to place these names into macro variables for subsequent processing can be effective. For example, the following call sets the macro variable Domains to the value contained in the DATA step variable DomNames:

```
call symput("Domains", strip(DomNames));
```

PROC SQL's INTO keyword

PROC SQL's INTO keyword is quite powerful for creating macro variables for automation. One of its common uses in automation is to provide a list of data sets to process. The following code does this:

```
proc sql noprint;
     select distinct memname into: DataSets separated by " "
     from dictionary.columns
     where libname eq "ANALYSIS";
quit;
```

In this case all distinct values of MEMNAME (data set name) from the LIBNAME called “ANALYSIS” are loaded into a macro variable named DataSets. The data set names in DataSets are separated by spaces. Subsequent macros, DATA steps, or SAS procedures could use &DataSets for analyses.

CALL EXECUTE

CALL EXECUTE is a powerful tool that enables users to call SAS macros and to execute DATA steps and SAS procedures from within DATA steps. This can be quite useful when automating assessment tools.

For example, assume there is a macro called %QCheck that checks QNAM/QLABEL variable pairs in a pair of STDM supplemental qualifier domains (perhaps a current domain and its previous version). A data set containing the names of all domains that contain QNAM/QLABEL variable pairs could be passed to a DATA _NULL_ step, and CALL EXECUTE could run %QCheck for each of the domains as follows:

```
data _null_;
     set SUPPDoms;
     call execute('%QCheck(domain=' || strip(memname) || ');');
run;
```

Although this example could be easily done using macro variables and a macro, CALL EXECUTE can create quite complex invocations, including dynamically generating and executing entire DATA and PROC steps. It can also be more straight-forward for less experienced SAS programmers to understand.

ALL, _NUMERIC_, and _CHARACTER_

Sometimes it is necessary to specify all the variables in a data set, all the numeric variables in a data set, or all the character variables in a data set without knowing (or caring) what the variables’ names are. The _ALL_, _NUMERIC_, and _CHARACTER_ keywords perform this function. For example, the following statements create two arrays: Num which contains all the numeric variables in a data set and Char which contains all the character variables:

```
array Num{*} _NUMERIC_;
array Char{*} $ _CHARACTER_;
```

ALL can also be used in conjunction with a LIBNAME’s name on the DATA= option of some SAS procedures, in order to specify all data sets in a library, as in:

```
proc contents data=SDTM._all_ ...;
```

BUILDING THE EXAMPLE TOOLS USING THESE COMPONENTS

This section describes the steps needed to create the examples presented in the Tools for Assessing Data section above. It also identifies how the components described above can be employed.

ASSESSING RELATED VARIABLES WITHIN DATA SETS

This is the simplest type of analysis tool described in this paper. It involves identifying the data sets that contain a pair or group of related variables (e.g. VISIT and VISITNUM), then seeing if these variables’ values are consistent within each data set. The specific programming steps are:

1. A PROC SQL step reads the COLUMNS DICTIONARY table, or a PROC CONTENTS step with *libname*._ALL_ on its DATA= option is used to get a list of all data sets and their variables.
2. A DATA step identifies data sets that contain the related variables of interest.
3. CALL EXECUTE is called once for each identified data set, to submit a PROC SQL step that finds all the distinct paired values of the related variables.
4. A DATA step identifies inconsistent mappings of the variables’ values within each data set, and it provides ODS formatting information to highlight any discrepancies.
5. PROC REPORT is used to present the results.

ASSESSING RELATED VARIABLES ACROSS DATA SETS

The sample tool described here checks VISIT/VISITNUM variable pairs across a set of SDTM domains. The specific programming steps are:

1. Steps 1 through 3 above are performed for VISIT and VISITNUM. Each resulting data set includes the name of its source domain and its VISIT/VISITNUM pair's values.
2. A DATA step merges these data sets by VISIT, checks for differences in each VISIT's VISITNUM values across all domains, and adds ODS formatting information to highlight any discrepancies.
3. PROC REPORT is used to present the results.

ASSESSING RELATED VARIABLES ACROSS TIME

The sample tool described here compares QNAM/QLABLE variable pairs' values within a set of SDTM supplemental qualifier domains (SUPP--) with the values found in a previous version of those same domains. It assumes the new domains are kept in one folder, and the previous version of the domains is kept in a separate folder. The specific programming steps are:

1. A PROC SQL step reads the COLUMNS DICTIONARY table to find all variables named QNAM or QLABEL in both the new and previous libraries (folders), where the data set name (MEMNAME) starts with "SUPP" and is six characters long.
2. A DATA step uses this information to identify domains that have both QNAM and QLABEL variables.
3. CALL EXECUTE is used to run a comparison macro for each of the identified domains. The comparison macro:
    a. Uses PROC SQL to find the distinct QNAM/QLABEL paired values in the new domain
    b. Uses PROC SQL to find the distinct QNAM/QLABEL paired values in the previous domain
    c. Uses a DATA step to merge these data sets by QLABEL, check for differences in each QLABEL's QNAM values, and add ODS formatting information to highlight any discrepancies
    d. Executes PROC REPORT to present the results

ASSESSING ALL DATA SETS AND VARIABLES OVER TIME

The sample tool described here assesses structural and data content changes that happen between two data transfers or snapshots. The resulting output is presented as a series of spreadsheets within a Microsoft EXCEL workbook. The exact details of the information shown in these spreadsheets was presented in a previous PharmaSUG paper (Stutzman, 2016).

The tool assumes that the new transfer/snapshot is kept in one folder, and the previous version is kept in a separate folder. The structural changes this tool identifies include:

- New or missing data sets
- New or missing variables in each data set
- Variable attributes that have changed

The data changes it evaluates include:

- Determining if the number of observations in each data set have increased or decreased
- Checking for new values in categorical variables
- Providing simple descriptive statistics for numeric variables in order to identify potential outliers

The specific programming steps are:

1. PROC CONTENTS steps gets data set and variable information for both the old and new snapshots.
2. PROC SQL steps creates two pairs of data sets: lists of data sets (and their record counts) in the old

and new, and lists of variables (and their attributes) in the old and new.

3. A pair of DATA steps merge the pairs of data sets just created. They look for and highlight mismatches between data sets and variables in the old and new snapshots, and they create data sets for subsequent reporting.
4. The ODS EXCEL XP tagset environment is set up, and an output EXCEL workbook (in XML format) is opened.
5. A PROC REPORT step creates a data sets level spreadsheet. It shows all data sets and the number of observations each data set contains. This spreadsheet and the ones created in the next two steps are based on the data sets created in Step 3 above.
6. PROC REPORT steps create two spreadsheets: one that shows all variables that only appear in the old snapshot, and one that shows all variables that only appear in the new snapshot.
7. A PROC REPORT step creates a spreadsheet that shows all variables that have attributes that differ between snapshots.
8. A DATA _NULL_ step reads the list of all variables that was created in Step 3 above. It uses BY processing to perform the following tasks for each data set in the new snapshot individually:
    a. A macro variable that contains a list of all numeric variables in the data set is created using CALL SYMPUT.
    b. After all records for a data set have been read, CALL EXECUTE is used to call a categorical values comparison macro for the data set. This macro:
        i. Uses ODS OUTPUT to get NLEVELS information from PROC FREQ for all variables in the old and new versions of the data set. NLEVELS determines how many unique values are found in each variable.
        ii. A DATA step merges the old and new NLEVELS data sets by variable name. It checks to see if the NVALUES values for each variable are less than a pre-specified threshold value. Variables that meet this criteria are assumed to be categorical, and a macro variable that contains all the categorical variables' names is created.
        iii. CALL EXECUTE runs a macro that compares all the categorical variables' values in the new snapshot with their values on the old snapshot. This macro accomplishes this by using ODS OUTPUT statements to get OneWayFreqs information from PROC FREQ steps that include all the categorical variables' names on their TABLES statements. There is one PROC FREQ step for the old snapshot and one PROC FREQ step for the new snapshot.

            The resulting OneWayFreqs data sets are merged by variable name and value in a DATA step. Values are compared, and ODS formatting information is added to highlight any discrepancies.
        iv. A PROC REPORT step creates a spreadsheet for the data set. It shows all categorical variables and their values.
    c. CALL EXECUTE is also used to call a macro that uses PROC MEANS, a DATA step, and PROC REPORT to create a spreadsheet that shows simple descriptive statistics for all numeric variables.
9. The XML workbook is opened in Microsoft EXCEL so it can be saved as an XLSX workbook.

ASSESSING CHRONOLIGICAL RELATIONSHIPS

The sample tool described here checks the chronological relationship between programs, their output, and their logs. It also includes information about test or quality control (QC) programs and their output and logs. Consistent folder structures and file naming conventions are essential for this type of assessment.

Before programming begins, a set of content and temporal rules should be established. Examples could include:

- Programs must have corresponding output and logs.
- A program's timestamp must precede the timestamps for its output and its log.
- A QC program's timestamp must come after the timestamp of the program it validates.

Once these rules have been established, a tool can be developed. The specific programming steps are:

1. Pipes are used to submit a series of Windows DIR commands (or corresponding commands, if in a non-Windows environment) to read all the relevant file system folders – gathering directory information for the all the files of interest: programs, output, and logs.
2. DATA steps parse the information form the FILENAMEs that established the pipes. The resulting data sets contain timestamp information for all items of interest. They also contain a common key value (perhaps the programs' output object names) that uniquely identifies related programs, QC programs, and their associated logs and output.
3. A DATA step merges the previously created data sets by the common key value, checks for deviations from the content and temporal rules that were developed beforehand, and adds ODS formatting information to highlight any discrepancies.
4. A PROC REPORT step presents the results.

CONCLUSION

The use of programmatic tools to assess data is an important part of ensuring consistency and quality. The more these tools can be automated, the more powerful and efficient these assessment processes become.

As mentioned previously, the goal was not to provide a comprehensive list of tools. The goal was to show some possibilities and to inspire other creative uses of the building blocks described herein.

REFERENCES

Black, Steven. 2011. "Excel Traffic Lighting and Street Paving Simplified." *Proceedings of the PharmaSUG 2011 Conference*. Nashville, TN: PharmaSUG. Available at http://www.pharmasug.org/proceedings/2011/TU/PharmaSUG-2011-TU03.pdf.

DelGobbo, Vincent. 2014. "Creating Multi-Sheet Microsoft Excel Workbooks with SAS®: The Basics and Beyond Part 1." *Proceedings of the SAS Global Forum 2014 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS050-2014.pdf.

DelGobbo, Vincent. 2015. "Creating Multi-Sheet Microsoft Excel Workbooks with SAS®: The Basics and Beyond. Part 2." *Proceedings of the PharmaSUG 2015 Conference*. Orlando, FL: SAS Institute Inc. Available at http://www.pharmasug.org/proceedings/2015/HT/PharmaSUG-2015-HT08-SAS.pdf.

Hadden, Louise. 2006. "STOP! WAIT! GO!: See What Traffic-Lighting Can Do For You!" *Proceedings of the SUGI 31 Conference*. San Francisco, CA: SUGI. Available at http://www2.sas.com/proceedings/sugi31/142-31.pdf.

Lafler, Kirk Paul. 2005. "Exploring DICTIONARY Tables and Views." *Proceedings of the SUGI 30 Conference*. Philadelphia, PA: SUGI. Available at http://www2.sas.com/proceedings/sugi30/070-30.pdf.

Sharma, Rajkumar. 2008. "CALL EXECUTE: A Hidden Treasure and Powerful Function." *Proceedings of the PharmaSUG 2008 Conference*. Atlanta, GA: PharmaSUG. Available at http://www.lexjansen.com/pharmasug/2008/po/PO11.pdf.

Stutzman, Paul. 2016. "Handling Interim and Incomplete Data in a Clinical Trials Setting." *Proceedings of the PharmaSUG 2016 Conference*. Denver, CO: PharmaSUG. Available at http://www.pharmasug.org/proceedings/2016/IB/PharmaSUG-2016-IB12.pdf.

Varney, Brian. 2008. “Check out These Pipes: Using Microsoft Windows Commands from SAS®.” *Proceedings of the SAS Global Forum 2008 Conference*. San Antonio, TX: SAS Global Forum. Available at http://www2.sas.com/proceedings/forum2008/092-2008.pdf.

CONTACT INFORMATION

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

Paul Stutzman
Axio Research, LLC
pauls@axioresearch.com