PharmaSUG 2015 – Paper QT28

Which TLFs Have Changed Since Your Previous Delivery? Get a Quick YES or NO for Each TLF

Tom Santopoli, Accenture, Berwyn, PA

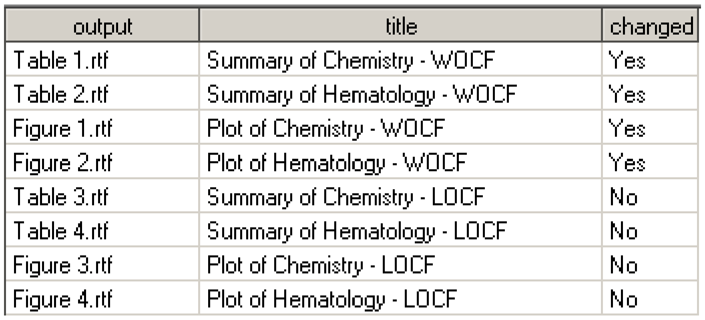
ABSTRACT

Ever need to make a delivery where only some of the TLFs are expected to have changed since the previous delivery? This paper presents a method to obtain a quick YES or NO for each TLF to indicate which TLFs have and have not changed since the previous delivery. Prior to investigating specific changes between TLFs, such as differences in lines and cells, a quick YES or NO for each TLF is a very useful overview to ensure that only the appropriate TLFs have been impacted by any updates. A quick YES or NO can be obtained even for Figures. A couple quick tips to obtain the results of PROC COMPARE and to use CALL EXECUTE for data-driven macro calls are presented as the primary coding techniques.

INTRODUCTION

Suppose the statistics displayed in a series of TLFs that present the results of the Worst Observation Carried Forward (WOCF) analysis are observed to be much higher than anticipated. A reviewer questions the results, leading to the discovery of a programming error in the calculation of WOCF. Last Observation Carried Forward (LOCF) is another analysis in the study. The variable DTYPE identifies the records to be used in the WOCF and LOCF analyses. In order to correct the WOCF analysis, an update is made to the derivation of DTYPE. After the update, DTYPE will identify a new group of records to be used in the WOCF analysis. However, the group of records DTYPE identifies for the LOCF analysis prior to the update should be the same group of records DTYPE identifies for the LOCF analysis after the update. The following results would give immediate peace of mind that there were no unintended consequences of the update to DTYPE:

Diagram 1



output	title	changed
Table 1.rtf	Summary of Chemistry - WOCF	Yes
Table 2.rtf	Summary of Hematology - WOCF	Yes
Figure 1.rtf	Plot of Chemistry - WOCF	Yes
Figure 2.rtf	Plot of Hematology - WOCF	Yes
Table 3.rtf	Summary of Chemistry - LOCF	No
Table 4.rtf	Summary of Hematology - LOCF	No
Figure 3.rtf	Plot of Chemistry - LOCF	No
Figure 4.rtf	Plot of Hematology - LOCF	No

The diagram above is a SAS® Data Set indicating which TLFs have changed. Only the TLFs that present the results of the WOCF analysis have changed, as expected. Of course, each of the TLFs associated with the WOCF analysis would then have to be thoroughly examined to verify that the values displayed in the body of these TLFs have decreased, as expected. The next section explains the coding techniques used to construct a Data Set similar to that in Diagram 1.

OBTAINING A QUICK YES OR NO FOR EACH TLF

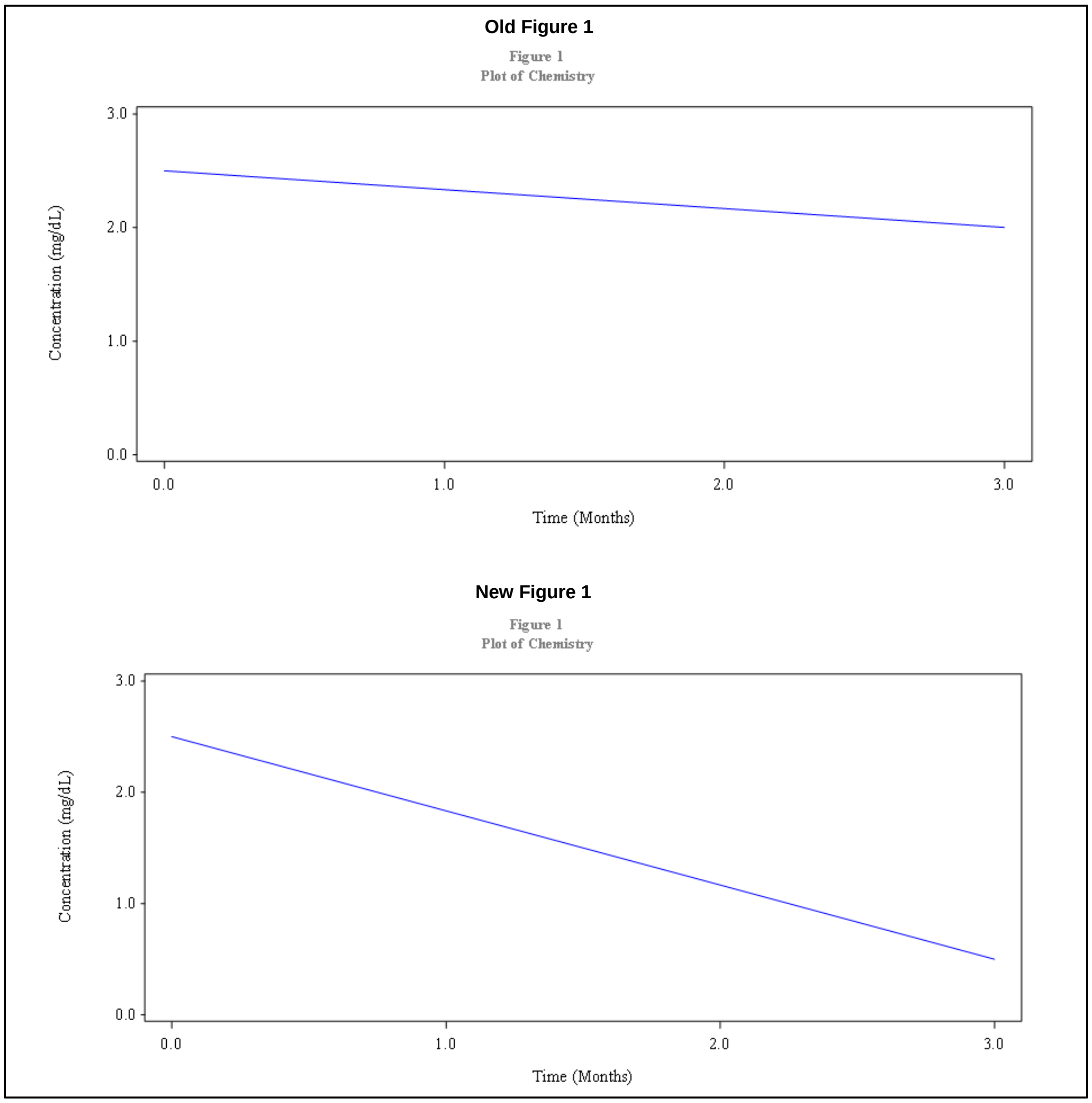
A Table of Contents can serve as the metadata to construct the Data Set in Diagram 1. If the Table of Contents exists in the form of an Excel spreadsheet, then the contents can be imported into the Data Set METADATA by using a PROC IMPORT:

```
proc import
        dbms=xls
        datafile="F:\Documentation\Table of Contents.xls"
        out=work.metadata replace;
        getnames=yes;
run;
```

The remainder of this section describes how the column that indicates changes to the TLFs is added to the Data Set METADATA. TLFs in RTF format are used as the example, but a similar concept can be applied to other types of files, such as XML, HTML, PDF, etc.

Lines of RTF code serve as the building blocks of RTF files. If two TLFs are identical, then the RTF code will be identical. If two TLFs are not identical, then there will be differences somewhere in the RTF code. A comparison of the RTF code can be used to determine if anything has changed. Differences in the body of a table, column headers, titles, footnotes, or even the shape of a figure will cause differences in the RTF code. Suppose the new version of Figure 1 must be compared to the old version of Figure 1:

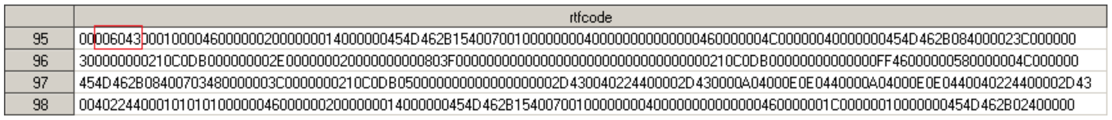
Diagram 2



The line in the new version of Figure 1 clearly takes a steeper drop than the line in the old version of Figure 1. The following is an excerpt of the same section of the RTF code for each version of Figure 1, with the differences enclosed in red boxes:

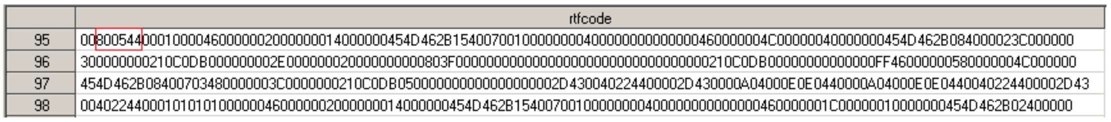
Diagram 3

Old Figure 1 RTF Code



	rtfcode
95	00006043000100004600000020000000140000000454D462B154007001000000004000000000000000460000004C00000040000000454D462B084000023C000000
96	300000000210C0DB000000002E00000002000000000803F000000000000000000000000000000000210C0DB00000000000000FF46000000580000004C000000
97	454D462B0840070348000003C0000000210C0DB0500000000000000000002D430040224400002D430000A04000E0E0440000A04000E0E0440040224400002D43
98	00402244000101010100000046000000200000001400000454D462B154007001000000004000000000000000460000001C00000010000000454D462B02400000

New Figure 1 RTF Code



	rtfcode
95	00800544000100004600000020000000140000000454D462B154007001000000004000000000000000460000004C00000040000000454D462B084000023C000000
96	300000000210C0DB000000002E00000002000000000803F000000000000000000000000000000000210C0DB00000000000000FF46000000580000004C000000
97	454D462B0840070348000003C0000000210C0DB0500000000000000000002D430040224400002D430000A04000E0E0440000A04000E0E0440040224400002D43
98	00402244000101010100000046000000200000001400000454D462B154007001000000004000000000000000460000001C00000010000000454D462B02400000

The steeper drop of the line in the new version of Figure 1 caused the differences in the RTF code that are enclosed in the red boxes. Row 95, character positions 3 to 8 from left to right differ. In the old version of Figure 1, the RTF code in this location contains the string “006043”. However, in the new version of Figure 1, the RTF code in the same location contains the string “800544”.

A simple strategy to compare TLFs is to perform a PROC COMPARE after reading the RTF code into SAS Data Sets. The macro %TLFCOMP accomplishes that objective:

```
%macro tlfcomp(cfile=);

        * read in files to compare ;

        data tlfbase;
                infile "Folder A\&cfile" length=reclen;
                input @1 rtfcode $varying400. reclen;
                if index(rtfcode,'creatim')>0 then delete;
        run;

        data tlfcomp;
                infile "Folder B\&cfile" length=reclen;
                input @1 rtfcode $varying400. reclen;
                if index(rtfcode,'creatim')>0 then delete;
        run;

        * compare files ;

        proc compare base=tlfbase compare=tlfcomp listall;
        run;

        %if &sysinfo=0 %then %let changed=No;
        %else %let changed=Yes;

        * append results ;

        data temp;
                length changed $12;
                changed="&changed";
        run;
```

```
23          data changed;
24                 set changed temp;
25          run;

26    %mend tlfcomp;
```

The code above compares RTF files in Folder A to different versions of the same RTF files in Folder B. The INFILE statements in lines 4 and 9 read the RTF code into SAS Data Sets. The RTF file to be compared is specified by the parameter CFILE in the macro call. Note that the lines containing the time the RTF files were created are removed in lines 6 and 11. Any footnotes containing the datetime stamp and owner should also be removed. Then, the PROC COMPARE in line 14 compares the SAS Data Sets containing the RTF code.

There are plenty of ways one can get blindsided by PROC COMPARE (see Horstman and Muller, 2013). A simple method to determine if the results of PROC COMPARE are perfectly clean is to evaluate the automatic macro variable SYSINFO. If the results of the PROC COMPARE are perfectly clean, then the value of SYSINFO will be 0 immediately after the PROC COMPARE. If the results of the PROC COMPARE are not perfectly clean, then the value of SYSINFO will be greater than 0 immediately after the PROC COMPARE. Lines 5 and 10 set the length of the variable RTFCODE to 400 in both Data Sets because even a difference in length would impact the value of SYSINFO.

Lines 16 and 17 set the macro variable CHANGED accordingly, based on the value of SYSINFO. The DATA Step in lines 19 to 22 stores the value of the macro variable CHANGED in the variable CHANGED, and the final DATA Step in lines 23 to 25 appends the results to the Data Set CHANGED. The Data Set CHANGED is initialized outside the macro as follows:

```
data changed;
       stop;
run;
```

An efficient method to call %TLFCOMP is with CALL EXECUTE:

```
data _null_;
       set metadata;
       call execute(cat('%nrstr(%tlfcomp(cfile=',strip(output),'));'));
run;
```

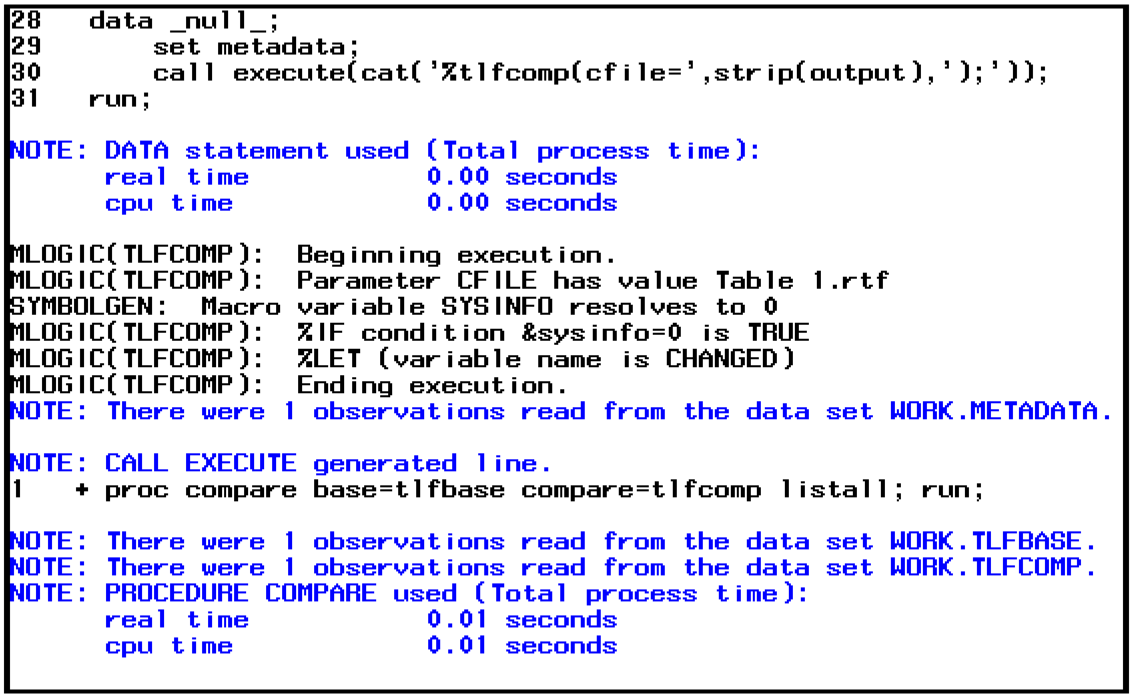
In this example, CALL EXECUTE is applied for data-driven macro calls. The CAT function assembles the call to the macro %TLFCOMP by concatenating the individual components that are separated by commas. The parameter CFILE is set for each value of the variable OUTPUT in the Data Set METADATA so that the macro %TLFCOMP can be called for each TLF being compared.

Note the use of the %NRSTR function in front of the call to %TLFCOMP. The need for %NRSTR highlights the mechanism by which CALL EXECUTE functions. When the argument to CALL EXECUTE resolves to a macro call, the following Steps take place:

1. DATA Step execution pauses and macro language statements execute IMMEDIATELY.
2. The remaining SAS code is placed on the input stack and executes AFTER the DATA Step has completed.

The purpose of %NRSTR is to mask the & and the %, which prevents SYSINFO from resolving and getting evaluated in Step 1. Without the application of %NRSTR, SYSINFO would resolve and get evaluated prior to the PROC COMPARE, as shown in the Log:

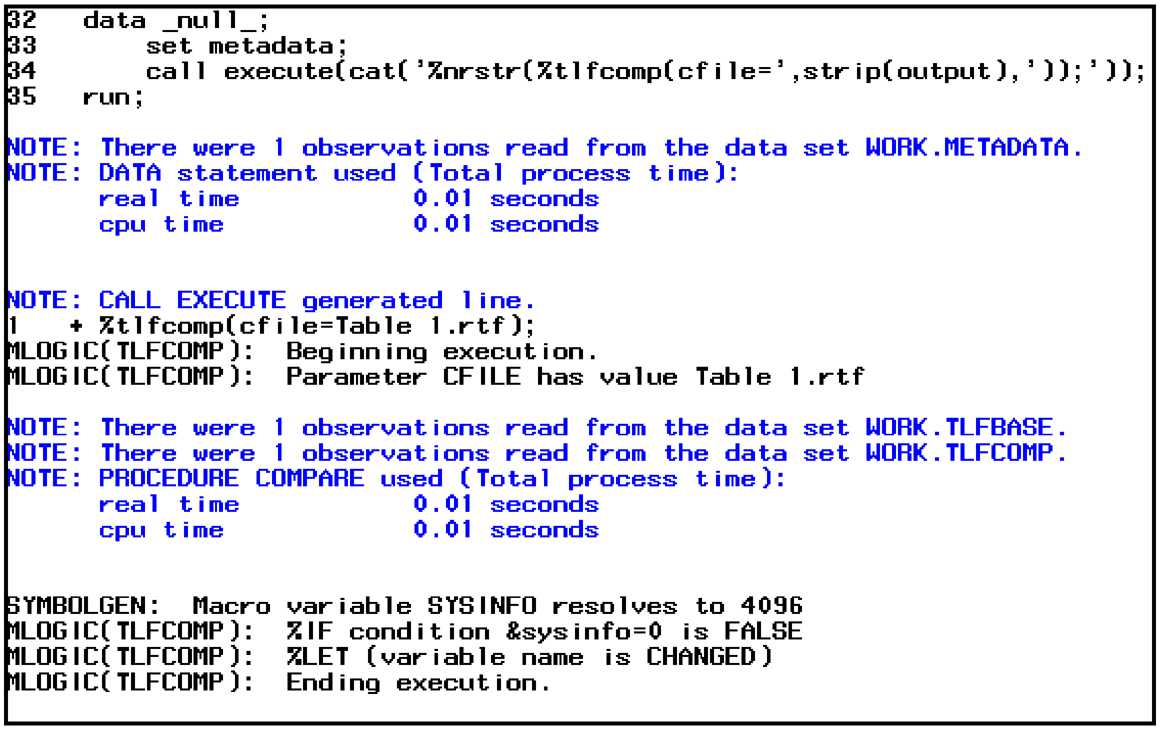
Diagram 4



```
28    data _null_;
29        set metadata;
30        call execute(cat('%tlfcomp(cfile=',strip(output),');'));
31    run;

NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

MLOGIC(TLFCOMP):  Beginning execution.
MLOGIC(TLFCOMP):  Parameter CFILE has value Table 1.rtf
SYMBOLGEN:  Macro variable SYSINFO resolves to 0
MLOGIC(TLFCOMP):  %IF condition &sysinfo=0 is TRUE
MLOGIC(TLFCOMP):  %LET (variable name is CHANGED)
MLOGIC(TLFCOMP):  Ending execution.
NOTE: There were 1 observations read from the data set WORK.METADATA.

NOTE: CALL EXECUTE generated line.
1    + proc compare base=tlfbase compare=tlfcomp listall; run;

NOTE: There were 1 observations read from the data set WORK.TLFBASE.
NOTE: There were 1 observations read from the data set WORK.TLFCOMP.
NOTE: PROCEDURE COMPARE used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

An accurate indication of the result of the comparison could not be obtained. Though the files compared are not equal, SYSINFO resolves to 0. However, with the application of %NRSTR, SYSINFO does not resolve and get evaluated until Step 2 at its position in the code after the PROC COMPARE, as shown in the Log:

Diagram 5



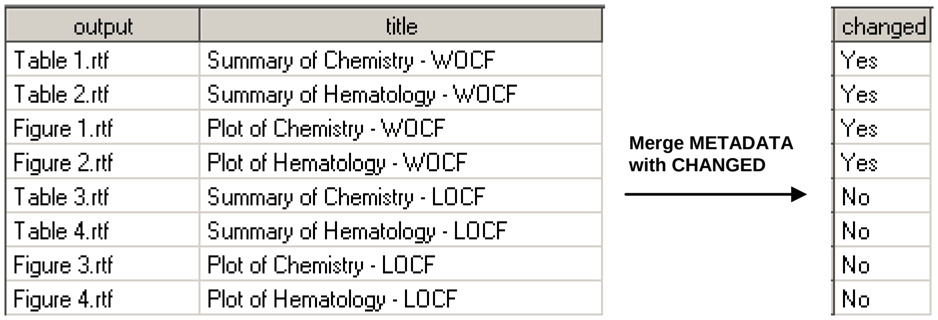
```
32    data _null_;
33        set metadata;
34        call execute(cat('%nrstr(%tlfcomp(cfile=',strip(output),'));'));
35    run;

NOTE: There were 1 observations read from the data set WORK.METADATA.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


NOTE: CALL EXECUTE generated line.
1    + %tlfcomp(cfile=Table 1.rtf);
MLOGIC(TLFCOMP):  Beginning execution.
MLOGIC(TLFCOMP):  Parameter CFILE has value Table 1.rtf

NOTE: There were 1 observations read from the data set WORK.TLFBASE.
NOTE: There were 1 observations read from the data set WORK.TLFCOMP.
NOTE: PROCEDURE COMPARE used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds


SYMBOLGEN:  Macro variable SYSINFO resolves to 4096
MLOGIC(TLFCOMP):  %IF condition &sysinfo=0 is FALSE
MLOGIC(TLFCOMP):  %LET (variable name is CHANGED)
MLOGIC(TLFCOMP):  Ending execution.
```

An accurate indication of the result of the comparison can now be obtained. The files compared are not equal, so SYSINFO resolves to a value greater than 0.

The results can be presented in the Data Set RESULT by merging the Data Set METADATA with the Data Set CHANGED:

```
data result;
       merge metadata changed;
run;
```

Diagram 6



output	title
Table 1.rtf	Summary of Chemistry - WOCF
Table 2.rtf	Summary of Hematology - WOCF
Figure 1.rtf	Plot of Chemistry - WOCF
Figure 2.rtf	Plot of Hematology - WOCF
Table 3.rtf	Summary of Chemistry - LOCF
Table 4.rtf	Summary of Hematology - LOCF
Figure 3.rtf	Plot of Chemistry - LOCF
Figure 4.rtf	Plot of Hematology - LOCF

changed
Yes
Yes
Yes
Yes
No
No
No
No

Some TLFs listed in the Table of Contents may not exist in both folders that contain the versions of the TLFs to be compared. The FILEEXIST function can be used to determine if a file exists in a folder:

```
%let exist=%sysfunc(fileexist(Folder A\Table 1.rtf));
```

In this example, the macro variable EXIST would be assigned a value of 1 if TABLE 1.RTF exists in Folder A. The %IF-%THEN-%DO statement can be applied in %TLFCOMP to circumvent the code if a TLF does not exist in both folders. The variable CHANGED can be set to NOT COMPARED instead of YES or NO.

CONCLUSION

The main feature of the report created by the techniques presented in this paper is the ability to visualize the names and titles of all the TLFs as they appear in a Table of Contents. This format is very conducive to identifying trends in which TLFs have and have not changed from one version to the next across a study. The techniques presented also open the door to using SAS for the next phase of the comparison, which is to determine exactly what has changed inside the body of the TLFs. In addition, the concept of applying CALL EXECUTE to call macros using prepared metadata as input can be implemented in the design of larger applications.

REFERENCES

Joshua Horstman and Roger Muller. *Don't Get Blindsided by PROC COMPARE.* Proceedings of the 2013 Pharmaceutical Industry SAS Users Group (PharmaSUG) Conference.

ACKNOWLEDGMENTS

I would like to thank Timothy Young and Wayne Zhong for providing valuable feedback in the development of the concepts discussed in this paper.

CONTACT INFORMATION:

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

Tom Santopoli
Accenture
1160 W. Swedesford Rd. Bldg. One, Berwyn, PA
(610) 407-7583
thomas.r.santopoli@accenture.com
www.accenture.com