PharmaSUG2012 - Paper CC08

Automation of Comparing ODS RTF Outputs in Batch using VBA and SAS ®

Dongsun Cao, UCB BioSciences, Inc., Raleigh, NC

ABSTRACT

In the pharmaceutical industry, Table, Listing and Graphs (TLGs), often produced in Rich Text Format (RTF), are the integral part of the new drug application (NDA) for approval to the regulatory agency. Therefore, it is essential to have an efficient quality check mechanism in place to ensure the quality of those outputs. One of the quality checks processes widely adopted is to have two programmers to perform independent programming and then visually check one RTF output against another. Because visual check is time consuming and error prone, it is important to have this process automated. So far, there are several papers published to address the automation but none of them implement a process without either changing original table structure or losing some information such as titles, footnotes or headings, thus limiting its use. In this paper, I present an automated process in which RTF outputs from two directories are compared against each other and comparison results are reported. The techniques involve using Visual Basic Application (VBA) macro to convert RTF outputs into text files which are then read into SAS and compared against each other using Proc Compare procedure. The described automated process will ensure that double programming scheme is religiously and efficiently implemented and should be able to add another layer of quality control to the TLGs production.

TLGs (Tables, Listings, and Graphs), Visual Basic for Applications (VBA), QC (Quality Check), ODS (Output Delivery System). RTF (Rich Text Format)

INTRODUCTION

Statistical reports are produced preferably in the RTF format in clinical trial industry. To ensure quality of those outputs, it is common to adopt double programming. Visual check of the TLGs from production side against QC has been often used. Although this type of quality control is a necessary step in the development of TLGs at the initial stage, it is essential to automate RTF comparing process when a large amount of output files or repeated reruns are involved. The major obstacle of automating the process has been conversion of RTF output files to the SAS datasets.

One strategy to automate the comparison of RTF outputs between two independent sources is that one side outputs SAS datasets that are about to be used in reporting for another to compare. This method is simple and easy. But the dataset structure and format have to be agreed upon in advance, requiring communications between two supposedly independent programmers and thus may compromise the validation process. Meanwhile, not all information in the RTFs are outputted into the datasets, for example, titles and footnotes.

A more advanced strategy involves direct comparisons of RTF files. With this approach, RTFs are first converted into SAS datasets and then comparisons are performed. This approach has significant advantages over the above mentioned one. First, this approach can potentially converts all RTF contents into a SAS dataset, including titles, footnotes, headings, and table cell contents. So, the automated comparison is presumably more complete. Secondly, since both production and QC programmers create the RTF outputs based on a common document, i.e. Table Shell, their RTFs should be identical if validation is passed. So this process will exclude the need of communications between two programmers prior to start of programming activity. Thus the independence of validation process is more strictly implemented.

Over the past several years, at least three distinct methods have been reported in converting RTFs to SAS datasets. One approach reads a RTF output as text file into a SAS dataset and then within SAS strips off the RTF tags programmatically to only leave the table contents (Hagendoorn, M et al., 2006).The advantage of this method is that it can be performed within SAS alone. The down side is that the code is specifically targeted for the RTF files produced with ODS techniques. If the RTF files are generated using manually coded RTF tags, it would fail to extract table contents correctly. Also, it is prone to errors if extra RTFs tags are added by a programmer in the Proc Report. So, the code needs to be constantly updated to accommodate changes in the SAS code. The second method invokes Save function in Word Basic within SAS through Dynamic Data Exchange (DDE) interface to save a RTF file into a text file (Xu and Zhou, 2007). This method compares the RTFs in the text format without any coding to remove the RTF tags. But it would change the table structure if used to convert the ODS RTF files. Also, the table structure is disrupted in which titles and footnotes are not with table in the converted text files and the paginations are also changed. The third approach used VBA macro to run the comparison. The macro can be invoked within SAS (Franklin D, 2007). This method is very efficient but since it used the VBA function to compare, the end report is not user friendly and hard to read.

In this paper, I present a new automated process for batch comparisons of RTF outputs. In this method, a VBA macro is developed to convert the RTF outputs in a specified directory to text files regardless of how they are produced; i. e. using ODS or manually coded RTF tags. The converted text files can be read into SAS and compared using Proc Compare. Because RTF outputs are faithfully converted into text files, we can not only detect differences between table contents but also can uncover the differences in table headings, titles, and footnotes. This approach provides an improved, efficient tool for validation RTF outputs.

VISUAL BASIC APPLICATION (VBA)

VBA (Visual Basic for Applications) is a programming language created by Microsoft. It is built on the Visual Basic (VB) language to extend Microsoft application's functionality. As its name implies, this language underlie the several important Window's application, including Word, Excel, and Access. In contrast to VB, VBA has to be run within host environment application such as Word.

VBA use object model in the programming language to deal with Window's applications. Object is something that are identified by its properties and it methods. In Word, documents, templates, paragraphs, fonts and border are all examples of objects in the object model.

VBA is especially useful for clinical trial programming because SAS programmers are often encountered with tasks such as concatenation of multiple RTF outputs into one document, conversion of a RTF file into a text file, Those tasks can not be easily performed with SAS but can be achieved without a hitch with VBA. Recently, there has been growing interests in the use of VBA in SAS environment (Franklin, 2007, Zeng et al, 2010).

Running VBA macro from SAS® and WORD

VBA macro is a section of VBA codes that are saved in the application such as Word. VBA macros can be run in two ways within the applications. Suppose that you have saved a macro named as myMacro. The first method is that you can run this macro by opening Word and clicking on Tools > Macros > myMacro > Run. The second method is opening the macro in Visual Basic Editor by clicking on Tools > Macro> Visual Basic Editor. Then copy your macro codes into the code editor window and hit Tools > Run.

In addition to running VBA macro within application, VBA macros can also be invoked within SAS using DDE. Suppose that we have a macro getODSTable saved in a Word document named as VBA-getTable.doc.

The first step is to start Word application. Option NOXWAIT instructs SAS to close DOS command prompt window without having to type EXIT when the process is finished.

```
option noxwait;
%let rc=%sysfunc(system(start winword));
```

The sleep function allows enough time for Word to open.

```
data _null_;
   x=sleep(3);
run;
```

The second step is to establish communication between SAS and Word by specifying the Fileref via DDE triplet.

```
Filename word DDE 'Winword|System';
```

Then, we use data _Null_ step to pass to word via DDE the word command to open desired Word file. and run the macro getODSTable and close the file.

```
data _null_;
   file word;
   put '[FileOpen .Name="C:\Users\Temp\VBA-getTable.doc"]';
   put '[getODSTable$()]';
run;
quit;
```

The final step is to close the Word application.

```
data _null_;
   file word;
   put '[FILEEXIT]';
run;
```

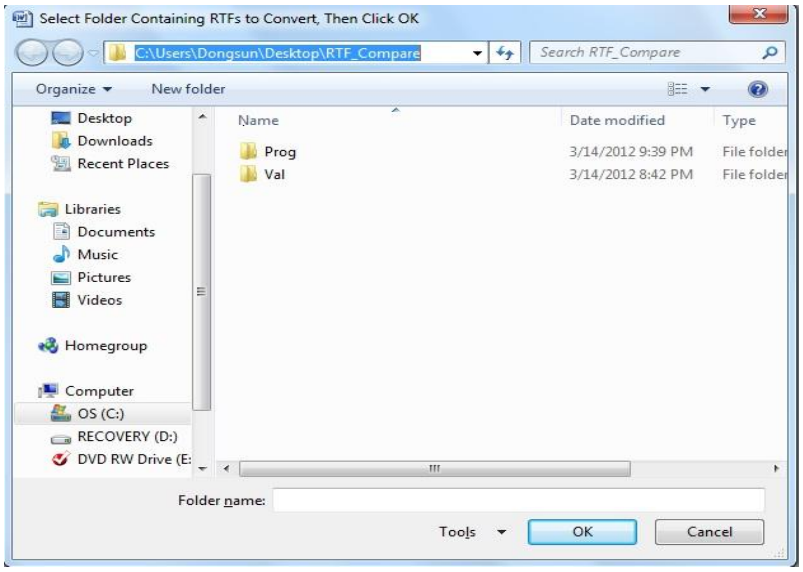
IMPLEMETNTAION

The automated RTF comparing process presented here consists of two steps. The first step is to run VBA macro getODSTable that converts RTF files from the production and QC sources into plain text files. The second step is to run a SAS macro RTF_Compare that reads text files into SAS, perform comparisons, and report the results.

VBA MACRO TO CONVERT RTF OUTPUTS IN A DIRECTORY TO TEXT FILES

Codes of the VBA macro getODSTable is listed in the Appendix. As mentioned above, you can either copy codes into Visual Basic editor in a Word document to run or invoke it from SAS through DDE. Using DDE has advantage of combining two steps of the program into to one and only SAS is needed. However, invoking Word macro from SAS is not as efficient as running the code from Word application directly, especially when there are large amount of RTF files involved. Therefore, it is recommended to run the VBA macro within Word.

The getODSTable is designed to allow you choose which directory you want to work from. After you copied the codes into Visual Editor and hit the Run button, it will first pop up a window that allows you to select the directory where RTFs reside in as shown below.


Select Folder Containing RTFs to Convert, Then Click OK

C:\Users\Dongsun\Desktop\RTF_Compare

Name	Date modified	Type
Prog	3/14/2012 9:39 PM	File folder
Val	3/14/2012 8:42 PM	File folder

Display 1. A pop-up window that allows a user to select a directory.

Then you select the folder where the RTF files are stored and click OK. The program will convert all the RTF files into text files. At the end of run, it will remind you with a window, stating that conversion is complete. The macro is also designed to handle some exceptions. For example, if you choose not to proceed after you hit the RUN, you can click Cancel button. Running would then be aborted and a window would pop up, stating that, running is cancelled by user (Figure not shown).

Now, the text files are generated and by default are stored in the same folder as the RTF files. But, target location of text files could be changed by modifying codes.

The text files have the following features:

- Only table contents will be extracted and converted,
- The table cells are delimited with Tab character to facilitate subsequent SAS import,
- At start of each page of output, a word ‘Page’ is inserted in order to mark the page start for later SAS processing to create a page number.

An example of a text file output is shown below. The left table is the sample table. The right table is the converted text file. It should be noted that if the table converted from ODS files may contain unexpected non-printing characters.

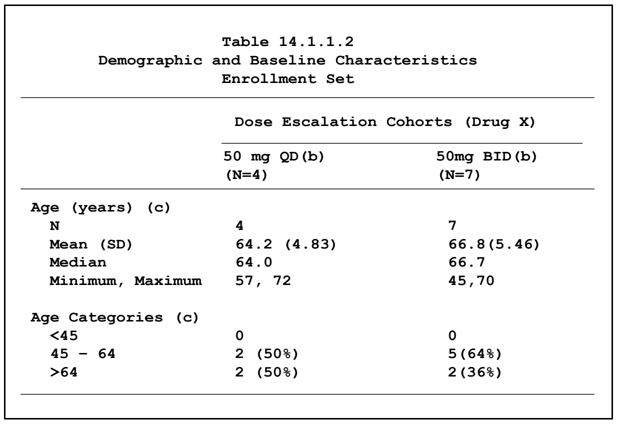
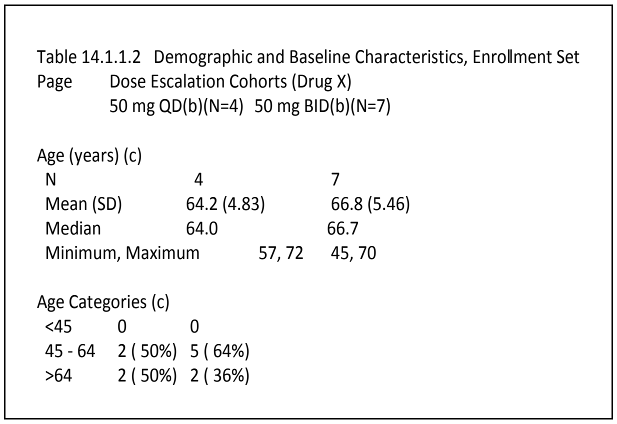


Table 14.1.1.2
Demographic and Baseline Characteristics
Enrollment Set

	Dose Escalation Cohorts (Drug X)	
	50 mg QD(b) (N=4)	50mg BID(b) (N=7)
Age (years) (c)		
N	4	7
Mean (SD)	64.2 (4.83)	66.8(5.46)
Median	64.0	66.7
Minimum, Maximum	57, 72	45,70
Age Categories (c)		
<45	0	0
45 – 64	2 (50%)	5(64%)
>64	2 (50%)	2(36%)



```
Table 14.1.1.2  Demographic and Baseline Characteristics, Enrollment Set
Page       Dose Escalation Cohorts (Drug X)
           50 mg QD(b)(N=4)  50 mg BID(b)(N=7)

Age (years) (c)
 N                  4                 7
 Mean (SD)         64.2 (4.83)       66.8 (5.46)
 Median            64.0              66.7
 Minimum, Maximum           57, 72    45, 70

Age Categories (c)
 <45        0          0
 45 - 64    2 ( 50%)   5 ( 64%)
 >64        2 ( 50%)   2 ( 36%)
```

Output 1. Sample table to be converted (left) and converted text file (right).

MARCRO TO COMAPRE TEXT FILES AND REPORT.

Once RTF files are converted into text files, they can be read into SAS datasets for further processing. Below is the brief description of key components of SAS macro RTF_Compare.

- **Get the list of file names that are common to both directories**

The first step is to get file names in the both directories that are going to be compared. Below code extracts the filename with extension TXT.

```
    filename bFolder "&bfolder";

    data bFile(keep=file fname fext cnt);

       length file fname $50;

       did=dopen('dir');
       cnt=0;
       do i=1 to dnum(did);
          file=dread(did,i);
          fname=scan(file, 1, '.');
          fext=upcase(scan(file, 2, '.'));
         if index(upcase(fext), 'TXT') then do;
             cnt +1;
             output;
         end;
       end;
       rc=dclose(did);
   run;
```

Likewise, a dataset cFile can be created to contain the file, filename and extension for QC directory.

Next, we identify common files that exist in both directories and create macro variables assigned with each file name and total number of files.

```
    data commf;
       merge bFile(in=base)
             cFile(in=comp);
       by file;
      if ^(base and comp)put "File only exists in one of folders: -"   file=;
       if base and comp;
    run;

    data _null_;
       set commf end=last;
       call symput("file"||strip(put(_n_, 2.)), strip(file));
       call symput("fname"||strip(put(_n_, 2.)), strip(fname));
       if last then call symput("tot", strip(put(_n_, 2.)));
    run;
```

- **Reading text files into SAS datasets**

Each text file will be read into SAS dataset using Proc Import. As mentioned above, the VBA-converted text file using Tab ('09'x) as delimiter for table cells, so delimiter in the Proc Import should use the '09'x.

```
%let dlmi='09'x;

proc import out=temp datafile ="&infile" dbms=dlm replace;
    delimiter=%unquote(&dlmi);
    getnames=no;
run;
```

It is expected that tables in a study will have varying number of columns. To accommodate this variability, Proc Contents is used to generate a dataset about a table's metadata. Then two macro variables are created, one to be assigned with concatenated variable names that specify table columns, other is to hold the total number of columns in the table. These two variables will be used in the Proc Report.

```
proc contents data=temp noprint out=allvars;
run;

data _null_;
   length allvars $200;

   retain allvars ' ';
   set allvars end=eof;
   name =tranwrd(name, 'VAR', 'Col');
   allvars = trim(left(allvars))||' '||left(name);
   if eof then call symput('varlist', allvars);
   if eof then call symput('numvar', strip(put(varnum, 2.)));
run;
```

- **Comparison and report the discrepancy.**

Now the output datasets are ready to be used in Proc Compare. Comparison results are outputted and will be as an input dataset in Proc Report.

```
proc compare data=base&i comp=comp&i out=&&fname&i outbase outcomp outnoequal
   outdiff listall method=exact;
  id pg row;
  var &varlist;
run;

data &&fname&i(keep=pg row ftype col:);
   retain pg row ftype;
   set &&fname&i;
   length filename $50 ftype $20;
   filename="&&fname&i";

   if _type_="BASE" then do;
      ftypn=1;
      ftype="PROD";
   end;
   else if _type_='COMPARE' then do;
      ftypn=2;
      ftype="VAL";
  end;
  else if _type_='DIF' then do;
      ftypn=3;
      ftype="DIF";
  end;

  if _type_='DIF' then do ;
      array c(*) col:;
      do i=1 to dim(c);
```

```
            if verify(c[i], ".") =0 then c[i]="";
            if verify(c[i], 'X.')=0 then c[i]="XXX";
        end;
     end;
  run;
```

An example of the discrepancy report is shown below. As the table shows, DIF indicate a discrepancy. For instance, the title in the validation table has a typo. In row 6 on page 1, the column2 in the table has a difference because the validation table has a one more space between minimum and maximum.

Comparison of RTF Outputs between Production and QC for T14_1_1_2_DEMO

Page	**Row**	**Type**	**Col1**	**Col2**	**Col3**
	1	PROD	Table 14.1.1.2 Demographic and Baseline Characteristics, Enrollment Set		
		VAL	Table 14.1.1.2 Demographics and Baseline Characterstics, Enrollment Set		
		DIF	XXX		
1	4	PROD	Mean (SD)	64.2 (4.83)	66.8 (5.46)
		VAL	Mean (SD)	61.9(4.56)	65.1(5.30)
		DIF		XXX	XXX
	6	PROD	Minimum, Maximum	57, 72	45, 70
		VAL	Minimum, Maximum	58, 66	45, 70
		DIF		XXX	

Output 2. Comparison results between Production and QC.

CONCLUSION

A clinical trial typically generates a large number of outputs and involves many of re-runs during the process. An efficient and simple tool to QC those outputs are necessary to ensure the quality. The method presented in this paper has proven to be able to accurately identify any discrepancies in the table contents between two sources. Though two-step strategy is adopted, the process has been proven to be efficient. It takes about 4-5 minutes to have about 50 tables and listings compared. More importantly, this technique begins with RTF outputs directly and end product of the VBA macro maintains most of original layout, which includes the table contents and headings. It can not only be applied to ODS RTF outputs but also to the outputs generated by manually added RTF tags.

It should be pointed out that VBA macro presented here for extraction table contents only. But it can be enhanced to extract title and footnotes with little modifications. Thus, it could be used to compare the titles and footnotes between production and QC files or between files and Table Shell, which can add another layer of quality assurance.

REFERENCES

SAS/BASE® 9.2 User's Guide, The PRO IMPORT, CONTENTS, and REPORT Procedures.

David Franklin (2007) Comparing ODS RTF Output Files in Batch Using SAS (or, 500 ODS RTF Documents to Compare in 15 minutes!) , PharmaSUG 2007, TT02.

Hagendoorn, Michiel , Jonathan Squire and Johnny Tai. 2006. Save Those Eyes: A Quality-Control Utility for Checking RTF Output Immediately and Accurately. Proceedings of the Thirty-first SAS Users Group International Conference, paper 066-31..

Xu, Michelle and Jay Zhou. 2007. %DIFF: A SAS Macro to Compare Documents in Word or ASCII Format. Proceedings of the PharmaSUG 2007 Conference, Jonathan Squire paper CC09.

Zemin Zeng, Mei Li. (2010). Using SAS software and Visual Basic for application to Dynamically manipulate SAS List files. SAS Global Forum, paper 076-2010

ACKNOWLEDGMENTS

I would like to thank UCB BioSciences for consistently encouraging and supporting conference participation, and all UCB colleagues for their support. I sincerely would like to thank Dr. Jim Johnson and Christy Crump for their suggestions and making the participation in this year's conference possible.

CONTACT INFORMATION

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

Name: Dongsun Cao, Ph.D.
Enterprise: UCB BioSciences, Inc.
Address: 8010 Arco Corporate Drive
City, State ZIP: Raleigh, NC 27617
E-mail: ccc_82@hotmail.com
Web: www.UCB.com

APPENDIX

```
Sub getODSTable()

      Dim StrData As String, temp As String, myFile As String, DocName  As String, Location As String
      Dim MyRange As Range
      Dim i As Integer, j As Integer, k As Integer, intPos As Integer, oTbl As Table
      Dim fDialog As FileDialog

      Set fDialog = Application.FileDialog(msoFileDialogFolderPicker)
      With fDialog
          .Title = "Select Folder Containing RTFs to Convert, Then Click OK"
          .AllowMultiSelect = False
          .InitialView = msoFileDialogViewList
          If .Show <> -1 Then
             MsgBox "Cancelled By User"
             Exit Sub
          End If
          Location = fDialog.SelectedItems.Item(1)
          If Right(Location, 1) <> "\" Then Location = Location + "\"
      End With

      ChangeFileOpenDirectory Location
      myFile = Dir(Location & "*.rtf")
      If myFile = "" Then Exit Sub
      Do
          Set source = Documents.Open(FileName:=myFile, Visible:=True)
          source.Activate
          intPos = InStrRev(myFile, ".")
          DocName = Left(myFile, intPos - 1)
          StrData = ""
          Set MyRange = ActiveDocument.Range(0, 0)
          For i = 1 To ActiveDocument.Range.Information(wdActiveEndPageNumber)
             Set MyRange = MyRange.GoTo(What:=wdGoToPage, Name:=i)
             Set MyRange = MyRange.GoTo(What:=wdGoToBookmark, Name:="\page")
             With MyRange
                j = 0
                For Each oTbl In MyRange.Tables
                    j = j + 1
                    temp = oTbl.Cell(1, 1).Range.Text
                    oTbl.Cell(1, 1).Range.Text = "Page"
                    oTbl.ConvertToText vbTab, True
               Next
          temp = .Text & vbCr
          StrData = StrData & temp
           If j > 0 Then
            For k = 1 To j
               ActiveDocument.Undo
            Next
         End If
         End With
       Next i
       'End With
       Set MyRange = Nothing

       Open Location & DocName & ".txt" For Append As 1
       Print #1, Replace(StrData, vbCr, vbCrLf)
        Close #1
       Application.ScreenUpdating = True
       ActiveDocument.Close SaveChanges:=wdDoNotSaveChange
       myFile = Dir
     Loop Until myFile = ""
     MsgBox prompt:="Conversions Complete.", buttons:=vbInformation
End Sub
```