PharmaSUG2011 – Paper CC17

Combining RTF Graphs

Lucius Reinbolt, Celerion, Lincoln, Nebraska

ABSTRACT

ODS RTF easily enables the placement of SAS® graphical output from a single program into a document. While a multitude of graphical files can be produced, the real challenge is getting all those files into one integrated file for review or reporting use. Many existing concatenation approaches rely on manual labor or external programs. The process described in this paper uses a single SAS program to concatenate multiple individual RTF graphics files by adapting the literal RTF code. Combining all of these files within an efficient and versatile SAS environment provides an added layer of traceability when working in an industry compliant environment such as SAS Drug Development. Each individual rich text formatted file has a similar structure consisting of a header and document group area. The header group controls the display characteristics and the document group area defines the appearance of the display images. This SAS based concatenation method uses a common header group while expanding the document group area to include the graphical files that need to be added.

INTRODUCTION

ODS RTF allows for the output of SAS graphics to be presented in a rich text format that is readable by word processing applications such as Microsoft Word. A simple ODS statement put around a graphing procedure has the potential to create many graphics all put into a single reviewable document. The problem is that it is common to use many programs outputting many individual files. Opening a few files does not present much of a problem, but when a few grows into a lot this process can become cumbersome. Many methods are used to combine these graphics output into one document. The methods include manually combining the files or using a separate programming method based in an environment other than SAS. This requires the programmer to exit the SAS environment to combine and review SAS graphics. The method in this paper reads in the rich text formatted graphics files that are created by SAS graphing programs and combines them using a single SAS program. The entire process is performed in the SAS environment and the output is a report ready graphics file. The individual rich text files contain not only graphics, but rich text format code that allows word processing applications to interpret all things related to document presentation. A simple stacking of the individual RTF files will not allow each graphics file to be viewed. Instead an understanding of the RTF file structure is needed. Each file consists of two areas that need to be altered in order to allow a common file to be produced. These two areas are the header group and document group. The header group area defines display characteristics including fonts, colors, and style sheets and is consistent between similar figure documents. The document group area contains the actual figures, section breaks, and text being displayed. It has been shown that RTF files can be read in and reformatted so that the new file has the desired display characteristics [1]. The strategy with this method is to identify and keep the header information for the first RTF file providing a common header group area while creating a combined document group area containing all graphical files. A common trouble point is section breaks. Some documents have section break characteristics that are not compatible with a combined multiple figure document. These section breaks must be identified and updated to set the correct section break properties which will allow each graph to be displayed correctly on the page as it did in the individual files. The section breaks need to be inserted in between each RTF graphical file to ensure that the graphics will display as intended with respect to page orientation. Each RTF group area will be enclosed by braces ({}). This process will also identify which braces ({}) will be kept and which are removed so as to create a new document group area enclosed within a set of braces ({}) at the beginning and end of the combined area.

METHODS

This process reads in the individual RTF graphical files then identifies key areas and alters the formatting as necessary to create a common header group area and combined document group area. A simple macro call can make reading each file much easier and can look like the example code in sample code 1. Each RTF file is read in with a simple macro which also identifies key information to help process the concatenation RTF files into SAS data sets that can be combined in another step. The macro call states the location and file name as well as the order number of the file added and the total number of files being combined.

Sample Code 1: Example Macro Code

```
%macro add(in=,innum=,totalin=);
<data step programming>
%mend add;
%add(in=&outpath./figure-1.rtf, innum=1,totalin=6);
...
%add(in=&outpath./figure-6.rtf, innum=6,totalin=6);
```

A data step is used to read in the raw rich text formatted file and place it in a variable named ‘rtfcode’. Simple helper variables should also be created in a data step to help make the programming easier later on. These include the length of the RTF code and dummy order variables. A word editor program can help to read the raw RTF code. Sample code 2 contains an example of what might be seen.

Sample Code 2: Example RTF Code

```
\pard\plain\intbl\sb0\sa0\qc\f2\fs20\cf1{\b0\i0\f2\fs20\cf1 Figure 1  Mean (SD) Plasma
Analyte Concentrations Versus Time\cell}
{\row}
\pard\trowd\trkeep\trql\trgaph0
\cltxlrtb\clvertalt\cellx9380
\pard\plain\intbl\sb0\sa0\qc\f2\fs20\cf1{\b0\i0\f2\fs20\cf1  {\line (Linear Scale)}\cell}
{\row}
\pard{\par}
\sectd\linex0\endnhere\sbknone\headery1800\footery1440\marglsxn1440\margrsxn1440\margtsxn1800
\margbsxn1440
{\header\pard\plain\qc{
}}
{\footer\pard\plain\qc{
```

Identify the section break control words and determine if they need any alteration. Files with only one graph per file may not have the proper control words to allow for the proper display characteristics when viewed in a combined file. Section breaks will need to contain the ‘\sect’ control word which defines a new section. Files with only one figure may be missing this in the sequence of control words to define a section break. The data step SAS code that can correctly identify and update this section break code is as shown in sample code 3.

Sample Code 3: Section Break Correction Code

```
if index(upcase(compress(translate(rtfcode,"", "0123456789"))),"\SECT\") eq 0 and
  index(upcase(compress(translate(rtfcode,"", "0123456789"))),"\SECTD\LINEX\ENDNHERE\")
gt 0 then do;
    rtfcode=lowcase(tranwrd(upcase(rtfcode),"\SECTD","\SECT\SECTD"));
  end;
```

It is important to note that the key words may differ in their order or presence to describe the desired section break and the above code may be altered to accommodate those differences. The section break portion of each file should be identified and a copy of this code should be set aside in another data set. This section break will be added to the end of its parent file so as to set a definitive break before the next file. This will ease the transition between figure files with different display properties. Specifically, landscape and portrait settings will have different section break properties and it is important to correctly define the boundaries between files that have differing display characteristics. Next the program determines if it is necessary to find and update the section break line of code to place at the end of each individual RTF graphic file. The portion of code used to accomplish this is found in sample code 4. This program identifies the section break lines of code and saves this section to a separate data set to be added back in. It is important to note that the section break code may be on multiple lines. The example figures used have the section break on three lines of code as highlighted in sample code 2. Individual files also may have the control word to have no section break ‘\sbknone’. This control word needs to be removed at this point as a definite section break is needed.

Sample Code 4: Section Break Code

```
*create section break;
data break&innum.;
  set temp&innum.;
  if index(upcase(compress(translate(rtfcode,"", "0123456789"))),"\SECTD\LINEX\ENDNHERE\") gt
0 then found=ord;
  retain found;
run;
proc sort;
  by ord2 ord;
run;
data break&innum.;
  set break&innum.;
  by ord2 ord;
  if found ne .;
  if ord-found le 2;
run;
proc sort;
  by ord2 ord;
```

Sample Code 4: Section Break Code (cont'd)

```
run;
data break&innum.;
  set break&innum.;
  by ord2 ord;
  if _n_ le 3;
  rtfcode=tranwrd(rtfcode,"\sbknone","");
run;
```

The document section boundaries are defined by the braces ({}) around it. The braces must be removed in only the proper places to create a common document group area. This requires that the end brace (}) be removed from all but the last file added. The code to do this is found in sample code 5.

Sample Code 5: Ending Correction Code

```
*remove end bracket from all but last file.  this allows files to merge;
%if &innum.=%eval(&totalin.) %then %do;
data out&innum.;
  set temp&innum.;
run;
%end;
%else %do;
data out&innum.;
  set temp&innum.;
  by ord2 ord;
  if last.ord2 then rtfcode=substr(rtfcode,1,length-1);
run;
%end;
```

The beginning brace ({) issue can be resolved by removing the header group information in all but the first file along with a small portion of the document group. This is done by searching for key words towards the beginning of the document group section as seen in sample code 6. Removing the header section and some of the document section up to the line containing the control word 'WIDOWCTRL' effectively does this.

Sample Code 6: Header Section Code

```
*remove beginning information from all but first file.  this allows files to merge;
%if &innum.=1 %then %do;
data out&innum.;
  set out&innum.;
run;
%end;
%else %do;
data out&innum.;
  set out&innum.;
  by ord2 ord;
  if index(upcase(rtfcode),"WIDOWCTRL") gt 0 then found=_n_;
  retain found;
run;
data out&innum.;
  set out&innum.;
  if found ne .;
run;
```

Now add the correct section break to the end of each data set containing the corrected RTF code as seen in sample code 7. This step is not needed for the last file to be added. The resulting data sets are now ready to be combined and output as the combined RTF file.

Sample Code 7: Final RTF Cods Data Step

```
data out&innum.;
  %if &innum. ne %eval(&totalin.) %then %do;
    set out&innum. break&innum.;
  %end;
  %else %do;
    set out&innum.;
  %end;
run;
```

A slightly modified method of combining and outputting the RTF graphical files as described in [1] is used to read in all corrected RTF code data sets and then output the final file.

CONCLUSION

This paper presents a method of combining multiple SAS RTF graphics files where all concatenation is performed within the SAS environment. This is accomplished by reading in the raw RTF code and restructuring the individual rich text files so that they may be combined to form one readable document. The result is a single document containing SAS graphical output that is review ready. Keeping the concatenation within the SAS environment provides not only efficiency and versatility, but also adds in a layer of traceability to this process.

REFERENCES

1. Zhang, Lei. 2005. "Constructing Stack Tables With Proc Report and ODS RTF," *Proceedings of PharmaSUG '05*, Phoenix, AZ.

ACKNOWLEDGMENTS

The author wishes to thank colleagues Steve Kirby and Yuli Xie who gave their time to review and give valuable feedback.

CONTACT INFORMATION

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

Lucius Reinbolt
Celerion
621 Rose Street Lincoln, NE 68502
Work Phone: (402)437-1115
e-mail: lucius.reinbolt@celerion.com