SESUG Paper 071-2021

Page Margin Checking Macro for RTF files

Lily Zhang, Huei-Ling Chen, Lei Sun, Eric Qi
Merck & Co., Inc., Kenilworth, NJ, USA

ABSTRACT

The Rich Text Format (RTF) file is the most popular output file format for clinical study reports. Different companies may have different standards for the format of Tables, Listings, and Figures (TLF) in the pharmaceutical industry. One such standard would be a general table format (e.g., title, footnote, table width, text font, border, or borderless). Other standards include page formats, for instance, page orientation and page margins. It is crucial to set up margin standards on every side of the page to avoid obscuring information when printed and bound. Thus, a programmer needs to ensure page orientation and margins of all RTF files in a deliverable package meet the criteria. Manually checking each file is not very practical, so we derived a macro for the task. This macro is easy to use to check RTF files from one or multiple folders. This paper will demonstrate the technique utilized in this macro and introduce the key RTF syntax that related page orientation and page margins.

KEYWORDS

Rich Text Format (RTF), Page Margin, Page Orientation, SAS Component Language (SCL), CSR

INTRODUCTION

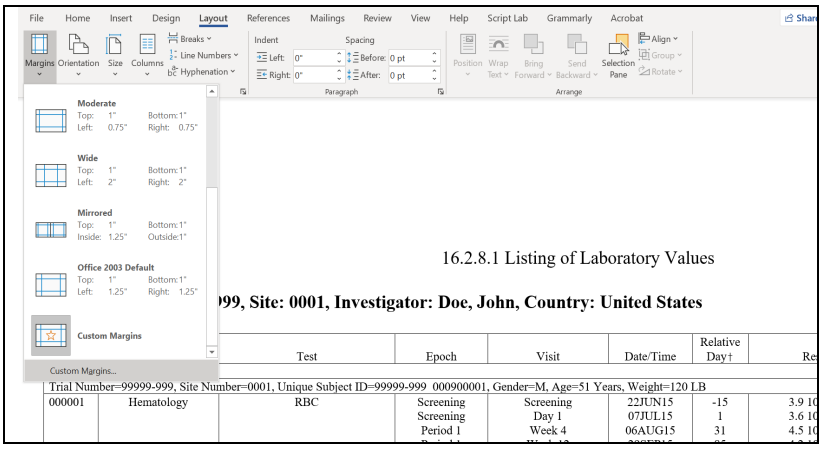
Many pharmaceutical and biotechnology companies use the RTF file format to submit clinical study reports (CSR) and other regulatory agency requirements. At our company, we have a set of standard worldwide margins to ensure that a document (e.g., CSR) is suitable for submission to all regulatory agencies and other requirements such as document binding, headers, and footers. Recently, we have a new update on tables, listings, and figures margin requirements. All standard, project/protocol-specific, and statistical results tables must be programmed according to these new table-formatting standards. The following table lists the new requirements for portrait and landscape margins.

Margins required for CSR tables (all numbers shown as inches)			
Portrait Margin		Landscape Margin	
Top	1.75	Top	2.0
Bottom	1.25	Bottom	1.25
Left	1.25	Left	1
Right	1	Right	1

How can we ensure every RTF file has completed the necessary update and page margins following the new criteria? A handy tool to quickly check whether RTF files meet the requirements would be desirable. Therefore, we derived a SAS macro %check0rtf0margins to check the margins and report the TLFs that do not meet the criteria.

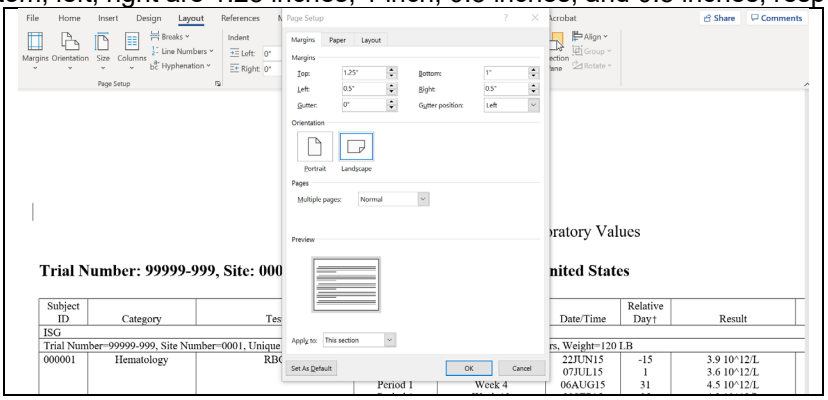
VIEW PAGE MARGINS IN MICROSOFT WORD MANUALLY

1. Open an RTF file in Microsoft Word.
2. Click on "**Layout**"
3. Click on "**Margins**"
4. Click on "**Custom Margins**…" at the bottom of the drop-down list.



Display 1: checking page margins of 'test.rtf' from Microsoft Word

5. The Page Margins would display in the pop-up window. In this example, 'test.rtf', the margins for top, bottom, left, right are 1.25 inches, 1 inch, 0.5 inches, and 0.5 inches, respectively.



Display 2: page margin values of the 'test.rtf'

This is how to check the page margins for an RTF file manually, but it's unrealistic to check all RTF files in this manner. A programmatical way to checking is to use the RTF syntax of page properties.

VIEW PAGE MARGINS IN SOFTWARE NOTEPAD

A way to view the original RTF code is to open an RTF file using the basic text-editing software Notepad.

Display 3: checking page margins of 'test.rtf' from Notepad

RTF SYNTAX OF PAGE PROPERTIES

RTF syntax is programmed using a backslash (\) and a control word to control features such as text font, table format, page properties, etc. For example, one commonly known RTF syntax to bold a text is **\b**, a backslash followed by a letter 'b'. A backslash followed by **\b0** is to make text bold off.

Below is the RTF syntax for the page properties. Similarly, it is a combination of the backslash and a control word.

RTF Syntax	Page Properties	Example
\margl	Margin left in twips	\margl720 (margin left is 0.5 inches)
\margr	Margin right in twips	\margr720 (margin right is 0.5 inches)
\margt	Margin top in twips	\margt1800 (margin top is 1.25 inches)
\margb	Margin bottom in twips	\margb1440 (margin bottom is 1 inch)
\paperw	Paper width in twips	\paperw15840 (paper width is 11 inches)
\paperh	Paper height in twips	\paperh12240 (paper height is 8.5 inches)

Table 1: RTF syntax for the page properties

Measurements in RTF are generally in '**twips**'. A twip is 1,440th of an inch. Inversely, one-inch equals 1440 twips. The "\margl720" means that the left margin has 720 twips, which is 0.5 inches (720/1440).

MACRO FUNCTION AND PARAMETERS

The purpose of Macro **%check0rtf0margins** is to check and summarize the page orientation and margins of RTF TLFs.

This macro consists of only four parameters:

1. **&folder_rtf_files** - the reference of RTF TLF's location path
   The input values should be a file reference (filename). This macro can recognize and process multiple file references, separated by a vertical bar '|'.
   For example, `folder_rtf_files=%str(folder1 | folder2 |floder3).`

2. **&output_excel_file** - the path and name of the output excel file
   The input value should be a file reference (filename) and output Excel file name within parenthesis.
   For example, `folder(rtf0margin0chk.xlsx).`

3. **&incorrect_margin_only_yn** - the parameter that allows the user to select to output TLFs with incorrect margin only or output all TLFs
   The valid value is "Y' or 'N'.

4. **&debug** - the parameter that allows the user to keep working data or not
   The valid value is "Y' or 'N'.

MACRO STRUCTURE

Step 1: Use the SAS Component Language (SCL) functions to save RTF output filenames to a dataset

To check multiple RTF files under specified folders, we need to extract RTF output filenames and save them into a dataset. We will use this dataset later to speed up the margins checking process.

Use the **FILENAME function** to assign **filrf** to an RTF output folder. Next, the **DOPEN function** is to open the directory identified by **filrf**. The **DNUM function** returns the total number of files in the RTF output folder. The **DREAD function** returns the name of each RTF file.

1. **Get the path of the RTF TLFs folder location.**

```
%let dirnum=%eval(%sysfunc(countc(&folder_rtf_files, '|' ))+1);

%do i=1 %to &dirnum;
   %let dir&i=%sysfunc(scan(&folder_rtf_files, &i, '|'));
   %let path&i=%sysfunc(pathname(&&dir&i,f));

   data _null_;
       rc1=filename("f1","&&path&i");
       RtfDirEx=dopen("f1");
       if RtfDirEx=1 then
            call symput("RtfRootDirValid"||put(&i,1.),"Y");
       else
            call symput("RtfRootDirValid"||put(&i,1.),"N");
   run;
%end;
```

2. **Open RTF TLF folders and saved the filenames to a dataset.**

```
data _outtable;
    rc=filename("filrf","&&path&j");
    did=dopen("filrf");
    if did > 0 then do;
        num=dnum(did);
        call symput ('fnum', num);
        do i=1 to num;
            if scan(dread(did,i),-1) = 'rtf' and
               index(dread(did,i),'$') eq 0 then do;
                filenm= dread(did,i);
                output;
            end;
        end;
    end;
run;
```

The following example is the dataset that contains four RTF filenames under a folder.

rc	did	num	i	filenm
0	1	4	1	discon0listing.rtf
0	1	4	2	disp0summary.rtf
0	1	4	3	swimpl.rtf
0	1	4	4	waterfl.rtf

Display 4: an example dataset that contains four RTF filenames under a folder

Step 2: Assign each RTF filename with individually corresponding macro parameters

Use PROC SQL procedure to assign the file name as macro parameter filename1, filename2, etc.

```
proc freq data=_outtable noprint;
     table filenm / out=a;
run;

proc sql noprint;
     select count(*) into :tot
```

```
     from _a
     ;
quit;

%let tot=&tot;

proc sql noprint;
     select filenm
     into :filename1 - :filename&tot
     from _a
     ;
quit;

%do i=1 %to &tot;
   %let file=&&path&j.&slash.&&filename&i.. ;
      . . .
%end;
```

Step 3: Read every RTF file property into a SAS dataset

Use "INFILE" with an "INPUT" statement to read the RTF file into the SAS dataset, we can retrieve RTF format information, such as page orientation, margin values, etc.

```
data rtf ;
     infile "&file" missover length=l end=lastobs lrecl=2000;
     input string $varying2000. l;
run;
```

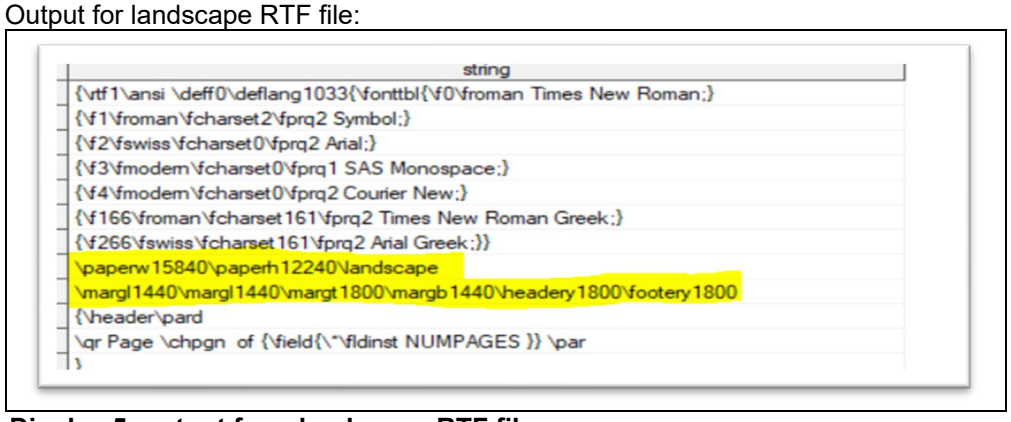
Step 4: Retrieve the page orientation and margin values (Left, Right, Top, Bottom)

The task to identify page orientation is to look into the numeric values attached with '**\paperw**' and '**\paperh**' in the RTF syntax. For landscape page orientation, the output data shows "**\paperw15840\paperh12240**" and/or "**\landscape**". The page size is defined with `\paperw15840\paperh12240` for a 11 inches wide and 8.5 inches high page. The page mode is defined with `\landscape` for landscape mode.

The portrait page orientation is the default setting of an RTF page. The RTF code either does not show page width and height information or has "**\paperw12240\paperh15840**".

Based on these settings, we use "**paperw**" and "**paperh**" to identify the page orientation.

Output for landscape RTF file:

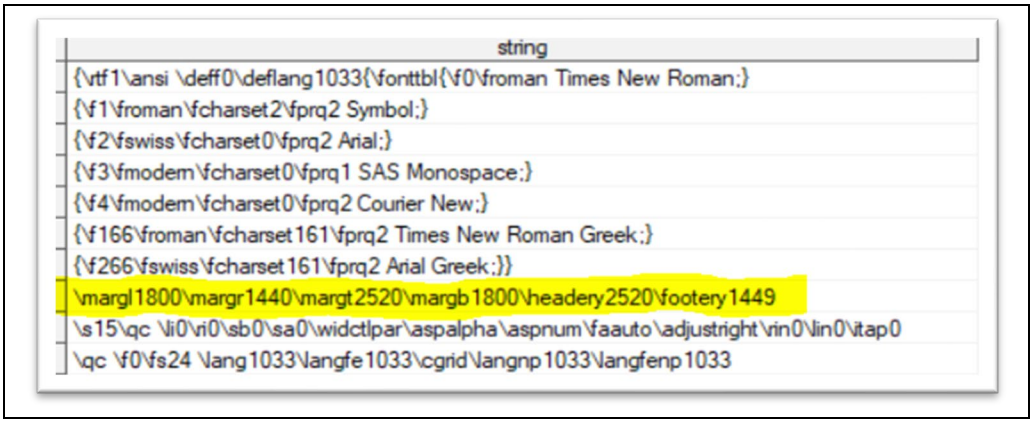


string
{\rtf1\ansi \deff0\deflang1033{\fonttbl{\f0\froman Times New Roman;}
{\f1\froman\fcharset2\fprq2 Symbol;}
{\f2\fswiss\fcharset0\fprq2 Arial;}
{\f3\fmodern\fcharset0\fprq1 SAS Monospace;}
{\f4\fmodern\fcharset0\fprq2 Courier New;}
{\f166\froman\fcharset161\fprq2 Times New Roman Greek;}
{\f266\fswiss\fcharset161\fprq2 Arial Greek;}}
\paperw15840\paperh12240\landscape
\margl1440\margl1440\margt1800\margb1440\headery1800\footery1800
{\header\pard
\qr Page \chpgn of {\field{*\fldinst NUMPAGES }} \par
}

Display 5: output for a landscape RTF file

```
if (index(string,"\paperw15840")>0 and
    index(string,"\paperh12240"))>0 then
```

```
        orientation="landscape";
    else orientation="portrait";
```

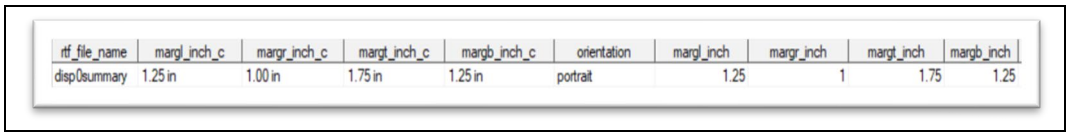


string
{\rtf1\ansi \deff0\deflang1033{\fonttbl{\f0\froman Times New Roman;}
{\f1\froman\fcharset2\fprq2 Symbol;}
{\f2\fswiss\fcharset0\fprq2 Arial;}
{\f3\fmodern\fcharset0\fprq1 SAS Monospace;}
{\f4\fmodern\fcharset0\fprq2 Courier New;}
{\f166\froman\fcharset161\fprq2 Times New Roman Greek;}
{\f266\fswiss\fcharset161\fprq2 Arial Greek;}}
\margl1800\margr1440\margt2520\margb1800\headery2520\footery1449
\s15\qc \li0\ri0\sb0\sa0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\qc \f0\fs24 \lang1033\langfe1033\cgrid\langnp1033\langfenp1033

Display 6: output for a portrait RTF file

Next task is to retrieve the margin values. We look into the numeric values attached with '`\margl`', '`\margr`', `\margt', and '`\margb`' in the RTF syntax to determine the page margin values in inches. Take Display 6 as an example, we learn that this RTF file has left margin 1800/1440=1.25 inch; right margin 1440/1440=1 inch; top margin 2520/1440=1.75 inch; and bottom margin 1800/1440=1.25 inch.

```
if index(string,"\margl")>0 and index(string,"\marglsxn") eq 0 then do;
   margl_start=index(string,"\margl");
   margl_string=substr(string,margl_start+1);
   margl_stop=index(margl_string,'\')-1;
   margl_inch=input(substr(margl_string,6,margl_stop-5),8.)/1440;
end;
if index(string,"\margr")>0 and index(string,"\margrsxn") eq 0 then do;
   margr_start=index(string,"\margr");
   margr_string=substr(string,margr_start+1);
   margr_stop=index(margr_string,'\')-1;
   margr_inch=input(substr(margr_string,6,margr_stop-5),8.)/1440;
end;
if index(string,"\margt")>0 and index(string,"\margtsxn") eq 0 then do;
   margt_start=index(string,"\margt");
   margt_string=substr(string,margt_start+1);
   margt_stop=index(margt_string,'\')-1;
   margt_inch=input(substr(margt_string,6,margt_stop-5),8.)/1440;
end;
if index(string,"\margb")>0 and index(string,"\margbsxn") eq 0 then do;
   margb_start=index(string,"\margb");
   margb_string=substr(string,margb_start+1);
     if index(margb_string,'\') > 0 then do;
       margb_stop=index(margb_string,'\')-1;
       margb_inch=input(substr(margb_string,6,margb_stop-5),8.)/1440;
     end;
     else do;
         margb_inch=input(substr(margb_string,6),8.)/1440;
     end;
end;
```



rtf_file_name	margl_inch_c	margr_inch_c	margt_inch_c	margb_inch_c	orientation	margl_inch	margr_inch	margt_inch	margb_inch
disp0summary	1.25 in	1.00 in	1.75 in	1.25 in	portrait	1.25	1	1.75	1.25

Display 7: output dataset with margin and page orientation information

Step 5: For margins not meeting the requirement, flag the record

A flag "Margin is not meeting criteria" will be marked in cases when an RTF file does not meet the criteria.

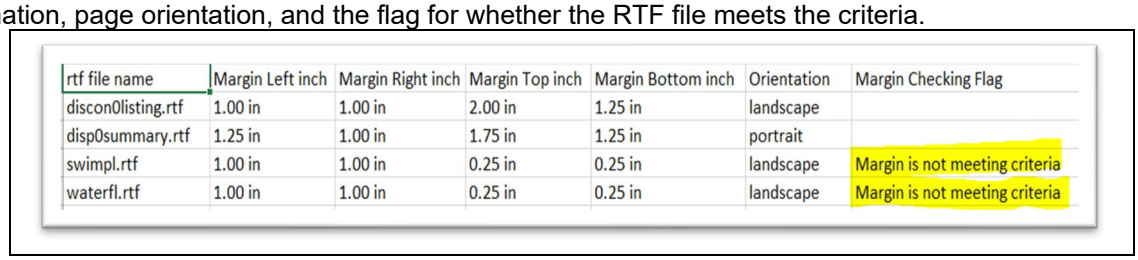
```
if orientation="portrait" then do;
   if margl_inch ne 1.25 or
      margr_inch ne 1 or
      margt_inch ne 1.75 or
      margb_inch ne 1.25 then margfl='Margin is not meeting criteria';
end;
if orientation="landscape" then do;
   if margl_inch ne 1 or
      margr_inch ne 1 or
      margt_inch ne 2 or
      margb_inch ne 1.25 then margfl='Margin is not meeting criteria';
end;
```

Step 6: Export the output to an excel file

```
proc export
   data= _final
   dbms=xlsx
   outfile="&excel_file_path.&slash.&excel_file_name"
   replace label;
run;
```

OUTPUT

This macro will generate an excel sheet that includes file name, left, right, top, and bottom margin information, page orientation, and the flag for whether the RTF file meets the criteria.



rtf file name	Margin Left inch	Margin Right inch	Margin Top inch	Margin Bottom inch	Orientation	Margin Checking Flag
disconOlisting.rtf	1.00 in	1.00 in	2.00 in	1.25 in	landscape	
disp0summary.rtf	1.25 in	1.00 in	1.75 in	1.25 in	portrait	
swimpl.rtf	1.00 in	1.00 in	0.25 in	0.25 in	landscape	Margin is not meeting criteria
waterfl.rtf	1.00 in	1.00 in	0.25 in	0.25 in	landscape	Margin is not meeting criteria

Display 8: output excel file

CONCLUSION

This paper presents a macro that can quickly check page margins of every RTF file in multiple RTF output folders. The main purpose is to ensure every RTF output has a consistent page margin regardless of the tables or graphs created from standard or project/protocol-specific macros. The macro is an efficient tool to help programmers to ensure page orientation and margins of all RTF files in a deliverable package meet the criteria when wrapping up projects.

REFERENCES

- https://www.fda.gov/files/medical%20devices/published/CTP-Electronic-Submission-File-Formats-and-Specifications.pdf
- SAS® Component Language 9.4: Reference, Third Edition

ACKNOWLEDGMENTS

The authors would like to thank Xiaohui Wang from Merck & Co., Inc., Kenilworth, NJ, USA, for her advice on this paper/presentation.

CONTACT INFORMATION

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

Lily Zhang
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: lily_zhang2@merck.com

Huei-Ling Chen.
Merck & Co., Inc., Kenilworth, NJ, USA
e-mail: huei-ling_chen@merck.com

Lei Sun
Merck & Co., Inc., North Wales, PA, USA
e-mail: lei_sun@merck.com

Eric Qi.
Merck & Co., Inc., North Wales, PA, USA
e-mail: eric_qi@merck.com

TRADEMARK