PharmaSUG 2022 - Paper QT-91

Auto Extraction of the Title Description in Table Report

Li Ma, Bo Zheng, and Xingshu Zhu, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

There are times during the completion phase of a study when programmers are asked to provide a full list of all table reports in Rich Text Format and their descriptive titles to help with preparing the Clinical Study Report or publications. When the number of total reports is small, this request is simple and can be quickly completed manually by opening each report and copying the title over. However, when there are hundreds of reports or more, it becomes a daunting task where a manual process is impractical and can dramatically increase the risk of mistakes. This paper shares an approach that utilizes RTF control words to programmatically identify the descriptive titles for a list of table reports and outputs the results to an Excel file that contains the report names, titles, and folder location links.

A report in RTF format that contains multiple tables is out of scope in this paper.

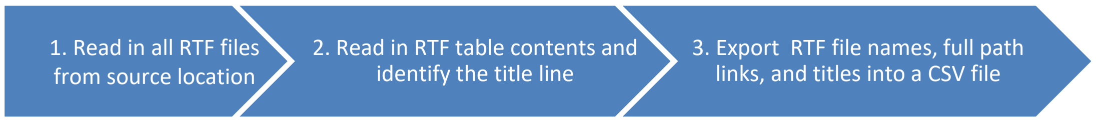
PC SAS version 9.4 (TS1M4) and Microsoft 365 MSO were used in this paper.

INTRODUCTION

In the pharmaceutical industry, analytical outputs are often created as tables and listings in Word .rtf file. There are no existing SAS procedures that can directly import a title from an RTF file into SAS and relevant files must be manually opened to copy their titles. This can be a tedious process even if the number of RTF files is relatively small and becomes administratively unmanageable when there are many. A customer request to change all the titles within a study report to a new format comes up quite often in RWE statistical programming and this process may require manually copying and pasting the title many times over and to varying places. This manual process is very inefficient and may cause errors.

In this paper, a SAS macro %read0rtf0title is introduced to read titles from .rtf documents in specified folder. It will import the entire contents of each RTF table into a SAS dataset and automatically identify the title line.

The general process to read titles out of RTF file is as follows:


1. Read in all RTF files from source location
2. Read in RTF table contents and identify the title line
3. Export RTF file names, full path links, and titles into a CSV file

METHOD

As the best practice, always check the availability of required parameters for the macro %read0rtf0title at the first section of the macro. Whenever a missing value is encountered on any of the required macro parameters, the macro will stop processing and send a customized error message to the SAS log window.

Parameter Name	Description	Default Value	Required (Y/N)	Valid Values	Comments
Indir	Input directory path from which .rtf files reside	NA	Y		The macro stop executing if the parameter is not provided
fileext	Specify file extension (i.e., rtf)	NA	Y		The macro stop executing if the parameter is not provided (the path in INDIR could have files in mixed extension)

filter	Specify the letters/keywords for files to exclude from output report	NA	N		For example, you have reports named like "tbl0**firstline**xx", and don't want them to appear in the macro-calling output, the FILTER parameter can be used to filter them out using "**filter=firstline**".
readtitle	Y is to read file Title, N or empty is not to read the file Title	NA	N	Y/N	If the files have no text-based titles, use N. (Example: The N option is used for a directory that only has RTF figure files with titles embedded in unreadable pictures. The names and links for these files are still created but the title will be blank.)
outpath	Output directory path	NA	N		If empty, an output .csv file with the finding results won't be created
outdat	Name used for temporary final output dataset and the .csv output file name	NA	Y		The macro stop executing if the parameter is not provided
debug	Delete all temp datasets if N; otherwise is Y.	N	N		If N, delete all temporary datasets at WORK lib except for the one in OUTDAT

Table 1. Macro parameters for read0rtf0title.sas

Here is a sample code used to check if OUTDAT parameter has a value provided when the macro is called. Customized error messages to call out the issue are made with %PUT, then use %GOTO EXIT to jump to the end of the macro (make sure there is the code line "%EXIT:" at the end of the macro):

```
%if %length(&outdat.)=0  %then %do;
  %put ========= Problem in Macro %upcase(&sysmacroname) call =======;
  %put;
  %put %str(ERR)OR: (&sysmacroname) OUTDAT: variable must be specified ;
  %put %str(ERR)OR: (&sysmacroname) Macro will not process your request.;
  %put;
  %put ========= Problem in Macro %upcase(&sysmacroname) call =======;
  %goto exit;
%end;
```

STEP 1: READ IN ALL RTF FILES FROM SOURCE LOCATION

Code to read in all RTF files from the directory &INDIR:

```
data filenm (keep=num file_name file);
  length file_name $50. file $40.;
    rc=filename('dir',"&indir");
    dirid=dopen('dir');
    do i=1 to dnum(dirid);
    file_name=dread(dirid,i);
    file=upcase(scan(file_name, 1, '.'));
      num=i;
      output;
    end;
    rc=dclose(dirid);
run;
```

Summary of SAS processes for the code above:

1. Create the dataset filenm that contains a list of file names under the directory &INDIR.
2. The dir that must be in single mark in DATA step is the specified FILEREF referring to external file location. The FILENAME function returns 0 if the operation was successful.
3. DOPEN opens a directory and returns a value greater than 0 that identifies an accessible directory.
4. DNUM function indicates the number of files in the given directory identifier (dirid). Do LOOP is used to read each file of the directory.
5. DREAD function returns the name of a file and save file name with extension.
6. Save the file name without extension.
7. DCLOSE function closes the directory &INDIR opened by the DOPEN function.

STEP 2: READ IN RTF TABLE CONTENTS ONE-BY-ONE, AND FIND THE TITLE LINE

The dataset filenm has a list of file names which were read from the &INDIR parameter.

There may be times when you only want the report names for files in the specified directory. The &READTITLE parameter provides the flexibility to get only report names or get both report names and report titles; its value of N means only reading report names, and Y to have both names and titles:

```
%if %upcase("&readtitle")="Y" %then %do;
```

Read in total count of report files into the macro variable TOTNUM_FILE and a list of report names into the macro variable FILE_NAMELIST that will be used in a DO LOOP later. Each report name is separated by "|" in the FILE_NAMELIST:

```
proc sql;
  select count(file_name) into: totnum_file
    from filenm
    where file_name ^= '';
  select file_name into: file_namelist
    separated by '|'
    from filenm
    where file_name ^= '';
quit;

proc sql noprint;
  create table final (title char(200), file_nm char(50));
quit;
```

Read each report file one-by-one within a DO LOOP. Get the report name using SCAN function, and then use INFILE statement to identify each report file at &INDIR (&DELIM is either "\" or "/" depending on the operating system your SAS environment runs on):

```
%do i_file = 1 %to &totnum_file.;
   %let eachfname=%scan(&file_namelist., &i_file., |);
```

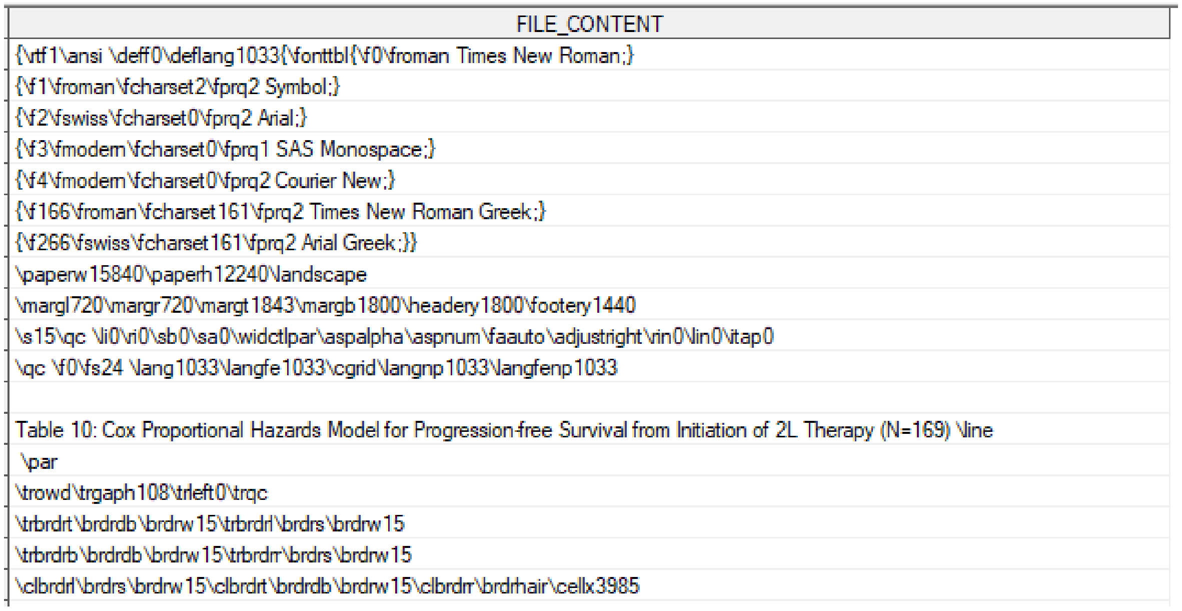
The contents of each report file get processed in the following codes. LINELEN is a value-unsigned variable that is created with the INFILE statement. When the first INPUT statement executes, SAS determines the line length of the record and assigns that value to the variable LINELEN.

This @ called trailing is a line-hold specifier. @1 moves the pointer to column 1 and reads each line of a RTF report into FILE_CONTENT variable. The LAG function puts the value read from INPUT statement to the temporary variable LAGFILE_CONTENT.

Then, every line of the RTF report is read into a separate observation in the STEP1 dataset, in the FILE_CONTENT variable:

```
data fcontent (keep=file_content);
  length file_content lagfile_content $ 200;
  infile "&indir.&delim.&eachfname." length=linelen;
  input @1 file_content $varying200. linelen;
  lagfile_content = lag(file_content);
  if _n_ eq 1 or file_content ne lagfile_content then output;
run;
```

The snapshot below shows an example of the FCONTENT dataset in part:



FILE_CONTENT
{\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
\margl720\margr720\margt1843\margb1800\headery1800\footery1440
\s15\qc \li0\ri0\sb0\sa0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\qc \f0\fs24 \lang1033\langfe1033\cgrid\langnp1033\langfenp1033
Table 10: Cox Proportional Hazards Model for Progression-free Survival from Initiation of 2L Therapy (N=169) \line
\par
\trowd\trgaph108\trleft0\trqc
\trbrdrt\brdrdb\brdrw15\trbrdrl\brdrs\brdrw15
\trbrdrb\brdrdb\brdrw15\trbrdrr\brdrs\brdrw15
\clbrdrl\brdrs\brdrw15\clbrdrt\brdrdb\brdrw15\clbrdrr\brdrhair\cellx3985

Display 1. Example of FCONTENT output dataset

One thing to mention is that to find the correct table description line, the description is required to start with the word “Table” (i.e., Table 10) that serves as a keyword identifier to allow the program to spot the table description line. If the RTF table report doesn’t have the keyword “Table” to start the title description, the title line will not be identified and read out by the macro, only the report’s name and full path will be saved to the CSV output.

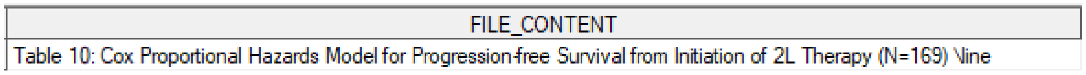
A table might have multiple pages and each page might have its own title line. When the total number of observations having the word “Table” is greater than two, only the first is kept:

```
data title_line;
  set fcontent;
  if index(upcase(file_content), 'TABLE') >0;
run;

data title_line;
  set title_line;
```

```
    if _n_ = 1;
  run;
```

The TITLE_LINE dataset only has one observation that is the title line, with or without the big N count (N=169 in the snapshot below):



FILE_CONTENT
Table 10: Cox Proportional Hazards Model for Progression-free Survival from Initiation of 2L Therapy (N=169) \line

Display 2. Example of a title with RTF control word \line

Note that in the above example, the RTF control word "\line" is a required line break with no paragraph break, it doesn't need to be shown in the final output result. Next is the code example for the TITLE_FINAL dataset:

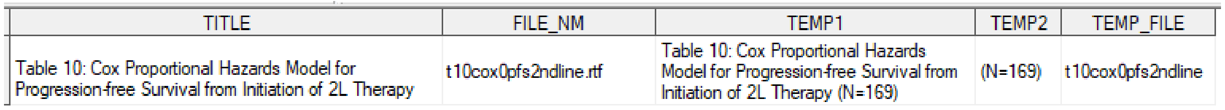
```
  data title_final (drop=file_content_);
    length title $200. file_nm $50. temp1 temp2 temp_file $200;
    set title_line (rename=(file_content=file_content_));

    if index(file_content_, '\line') >0 then do;
      temp1 = tranwrd(strip(file_content_), '\line', '');
      temp2 = scan(temp1, -1, ' ');
    end;
    else temp2 = scan(file_content_, -1, ' ');

    ** REMOVE BIG N, IF IT EXISTS IN THE TITLE OF A TITLE REPORT **;
    if index(temp2, '(N=') > 0 then do;
      temp_file = scan("&eachfname.", 1, '.');
      call symputx('n_'||temp_file, temp2);
      title = tranwrd(file_content_, symget('n_'||temp_file), '');
    end;
    else title = strip(file_content_);

    file_nm = "&eachfname." ;
    if index(title, '\line') >0 then title = tranwrd(strip(title), '\line', '');
    if strip(title) ^= '';
  run;
```

In the sample TITLE_FINAL dataset at display 3, the three TEMP prefixed variables are left in the dataset for demonstration purpose that hopefully reflects the codes to see how unwanted information (i.e., N=) on the title description line is picked out. The codes detecting the big N can be modified to fit your needs.



TITLE	FILE_NM	TEMP1	TEMP2	TEMP_FILE
Table 10: Cox Proportional Hazards Model for Progression-free Survival from Initiation of 2L Therapy	t10cox0pfs2ndline.rtf	Table 10: Cox Proportional Hazards Model for Progression-free Survival from Initiation of 2L Therapy (N=169)	(N=169)	t10cox0pfs2ndline

Display 3. Example row of final output dataset

The FINAL dataset template was created in an earlier step of the macro, it is attached by the title line in the TITLE_FINAL dataset at the end of DO LOOP. The title lines of all RTF reports in INDIR directory are identified by reading the contents of each RTF file in the DO LOOP codes section:

```
  ...
    data final;
      set final title_final;
    run;
  %end; /* end of DO LOOP */
```

STEP 3: EXPORT THE TITLES, ASSOCIATED RTF REPORT NAMES, AND REPORT FULL PATH INTO A .CSV FILE

Merge the FINAL dataset with FILENM dataset that was created at STEP1 and save the result to &OUTDAT and export to .csv file:

```
  data &outdat.;
      length file_path $500.;
      merge filenm
          final (rename=(file_nm=file_name));
        by file_name;

      file_path = "&indir.&delim." || strip(file_name) ;
    run;
  %end;
```

If the parameter &READTITLE=N, the DO LOOP is skipped and the macro jumps to the following code to save the RTF report name and full path into the final output &OUTDAT and exports it to a .csv file:

```
  %else %do;
      data &outdat.;
        set filenm ;
        length file_path $500.;
        file_path = "&indir.&delim." || strip(file_name) ;
      run;
  %end;

  %if %length(&outpath.)>0  %then %do;
      %if %upcase("&readtitle")="Y" %then %do;
        proc export data=&outdat.(keep=file_name file_path title)
          outfile="&outpath.&delim.&outdat..csv"
          dbms=csv replace;
        run;
  %end;
  %else %do;
        proc export data=&outdat.(keep=file_name file_path)
          outfile="&outpath.&delim.&outdat..csv"
          dbms=csv replace;
        run;
  %end;
%end;
...
```

CONCLUSION

This paper describes how to programmatically identify and output the titles of a list of RTF tables in three steps:

1. the entire content of RTF report files is read line-by-line into a SAS dataset
2. the table description line is automatically identified in the data steps by keyword
3. the report titles and their full path links are exported into a CSV file

The example macro brings the benefits of automation and avoids the potential human errors of doing this task manually. It also allows a standardized process for this task to be defined, which reduces the overhead resource costs of repeating this request multiple times during a project's life cycle. The output CSV report file can be expediently utilized by customers or other programmers to collaborate on creating a comprehensive Table of Contents with table names, titles, and full path of file location. Additionally, someone may utilize the macro and then extend the codes to compare a list of titles from two different folders in, for example, test and production environments to check for possible difference and report the error or warning message in case of any.

REFERENCES

Create a SAS Program that Can Read the Contents of a Directory

extension://elhekieabhbkpmcefcoobjddigjcaadp/viewer.html?pdfurl=https%3A%2F%2Fwww.lexjansen.com%2Fsesug%2F2018%2FSESUG2018_Paper-159_Final_PDF.pdf&clen=114671&chunk=true

RTF Control Codes/words list:

http://latex2rtf.sourceforge.net/rtfspec_62.html

ACKNOWLEDGMENTS

The authors would like to thank their management teams as well real-world evidence and epidemiology colleagues at Merck & Co., Inc., Upper Gwynedd, PA, USA, for their advice on this paper.

CONTACT INFORMATION

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

Li Ma
Merck & Co. Inc.
351 N Sumneytown Pike, North Wales, PA 19454, USA
li.ma2@merck.com

Bo Zheng
Merck & Co. Inc.
351 N Sumneytown Pike, North Wales, PA 19454, USA
bo.zheng1@merck.com

Xingshu Zhu
Merck & Co. Inc.
351 N Sumneytown Pike, North Wales, PA 19454, USA
xingshu_zhu@merck.com

TRADEMARK