PharmaSUG 2020 - Paper DV-198

r2rtf – an R Package to Produce Rich Text Format (RTF) Tables and Figures

Siruo Wang, Johns Hopkins Bloomberg School of Public Health, MD, USA;
Simiao Ye, Merck & Co., Inc., Kenilworth, NJ, USA;
Keaven M. Anderson, Merck & Co., Inc., Kenilworth, NJ, USA;
Yilong Zhang, Merck & Co., Inc., Kenilworth, NJ, USA;

ABSTRACT

The use of open-source R is evolving in drug discovery, research and development for study design, data analysis, visualization, and report generation in the pharmaceutical industry. The ability to produce tables, listings and figures (TLFs) in customized rich text format (RTF) using R is crucial to enhance the workflow of using Microsoft Word to assemble analysis results. We developed an R package, r2rtf, that standardizes the approach to generate highly customized TLFs in RTF format. The r2rtf package provides flexibility to customize table appearance for table title, subtitle, column header, footnote, and data source. The table size, border type, color, and line width can be adjusted in each cell as well as column width, row height, text format, font size, text color, alignment, etc. The control of the format can be row or column vectorized by leveraging the vectorization in R. Furthermore, r2rtf provides pagination, section grouping, multiple tables concatenations for complicated table layouts. In this paper, we provide an overview r2rtf workflow with examples for both required and optional easy-to-use functions. Code examples are provided to create customized RTF tables and figures with highlighted features. The open-source r2rtf R package is available at: https://github.com/Merck/r2rtf.

INTRODUCTION

Modern problems in medicine and pharmaceutical settings leverage open-source R to make new discoveries. While analyses are being conducted in the R platform, it is desirable to generate tables, listings, and figures (TLFs) in rich text format (RTF) using R. Later these TLFs can be directly inserted into a word document for the final analyses report generation. The ability to produce TLFs in customized RTF files using R is crucial to enhance the workflow of using Microsoft Word to assemble analysis results.

Although many tools and macros have been developed in SAS® programs over the past decades to generate RTF tables such as %SASWORD1 (Cunningham 1998), %SAS2WORD (Zhou 2001), %PRINT (Wehr 1996), %RTF (Peszek *et al.* 1998 and 1999), and particularly macro %RTFTable (Qi *et al.* 2003) and proc report in SAS ® that is very rich in functionality, there are only a few packages available to convert datasets to RTF tables with flexibilities in the R platform. The two existing R packages, rtf (Schaffer 2019) and gt (Iannone *et al.* 2020), currently provide limited ability to produce TLFs in RTF files for clinical reporting. The rtf package depends on two other R packages and provides 18 functions to handle different parts of the process. To generate TLFs, users are required to be familiar with the concept of RTF syntax with multiple functions such as start a paragraph, end a paragraph, add new lines, add a table, add text, set font size, etc. The gt package provides more flexibility to change table and figure appearance and present them in HTML, PDF and RTF. The gt package also depends on 14 other R packages and uses a large numbr of functions for detailed feature controls. Currently, the gt package is under active development and focuses on HTML format. Its ability to generate RTF tables is still under development.

Therefore, we are motivated to develop a new package, r2rtf, that provides easy-to-use and powerful functions to convert datasets into highly customized TLFs in RTF file using R. We carefully reduce the package dependency to simplify the qualification of the package for regulatory purpose. The only dependency of our r2rtf package is stringr (Wickham 2019) that is used for displaying special characters and symbols in latex code for the r2rtf package. The design of r2rtf is to reduce the efforts from users to set a large number of features for table formatting by given commonly used default settings (e.g., border type, color, line width, etc.). Compared to the existing rtf and gt packages, less dependency of other R packages makes r2rtf easy to install and ready for regulatory use.

Our package also minimizes the number of functions needed to generate TLFs which significantly enhances the workflow to generate TLFs in RTF files and reduce the user's input efforts. In detail, r2rtf provides three

required functions to generate a table body, four required functions to generate a figure body (two functions are shared with table body generation), and four optional functions to generate table titles, subtitles, column headers, footnotes, and data sources. With only 9 functions in total, r2rtf aims to maximize the flexibility for users to customize table and figure appearance by modifying parameter values. Compared to other existing packages, we provide more powerful and advanced features in page and table border related settings, text management inside and outside of tables, table concatenation, section grouping, and pagination.

The remainder of the paper is organized as follows. We start with an overview of the r2rtf package workflow on how to create tables and figures. Then we introduce functions and parameter values in four sections: (1) table body, (2) table column header, (3) title, subtitle, footnote, and data source, (4) figure generation. Finally, we illustrate some highlighted features of the r2rtf package, provide examples of function calls, present table outputs, and conclude.

PACKAGE OVERVIEW

In this section, we provide a front-end guideline for the r2rtf package, so users with programming skills at any level can easily follow the instructions to generate TLFs in RTF files using R. Because our package provides default values for most required parameters in the functions, users with basic R programming knowledge can easily call functions in r2rtf by providing minimal parameters to generate production ready TLFs. For users with advanced R programming knowledge, the r2rtf package provides optional parameters in function calls that users can take fully control to generate TLFs with highly customized appearance.

To generate a simple table, the r2rtf package provides three mandatory functions:

- `rtf_body()`: customizes the table body.
- `rtf_encode()`: extracts and renders table attributes to RTF encoding.
- `write_rtf()`: writes RTF encoding string to an RTF document.

In particular, `rtf_body()` function meets the user's need to take fully control of table appearance through parameters to customize table size, orientation, space, border type (e.g., single, double, dash, dot, etc.), color (e.g., 657 different colors named in `color()` function), line width, column width, row height, text format (e.g., bold, italics, strikethrough, underline and any combinations), font size, text color, alignment (e.g., left, right, center, decimal), etc. Format control can be at the cell, row, column, or table level. For example, parameter `text_justification = "c"` sets all text in the cells to be center adjusted and `text_justification = c("c", rep("l", 4))` sets the text in the first column to be center adjusted and the texts in the remaining 4 columns to be left adjusted for a table with five columns. In the following example, we use the three mandatory functions with minimal parameters to produce a simple RTF table in **Table 1** that includes a row of default column headers defined by variable names in the data set.

In this example, we first summarize the ADAE data set in the CDISC pilot project submission package (https://www.cdisc.org/pilot-project-submission-package) to a data set ready for table generation. The data set is called `ae_t1`. The first column in the data set lists the specific adverse events, and the remaining the 3 columns contain the number of subjects with specific adverse events in each treatment group. The definition code to produce **Table 1** is attached below. Command `%>%` from the dplyr (Wickham *et al.* 2019) package is used to pipe between functions:

```
ae_t1 %>%
  rtf_body(col_rel_width = c(3, rep(1,3)),
           text_justification = c("l", rep("c",3))) %>%
  rtf_encode() %>%
  write_rtf("rtf/ae_simple.rtf");
```

AEDECOD	Placebo	Xanomeline High Dose	Xanomeline Low Dose
APPLICATION SITE DERMATITIS	5	7	9
APPLICATION SITE ERYTHEMA	0	15	12
APPLICATION SITE IRRITATION	0	9	9
APPLICATION SITE PRURITUS	6	22	22
APPLICATION SITE VESICLES	0	6	0
BLISTER	0	0	5
COUGH	0	5	6
DIARRHOEA	9	0	5
DIZZINESS	0	12	8
ERYTHEMA	9	14	15
FATIGUE	0	5	5
HEADACHE	7	6	0
HYPERHIDROSIS	0	8	0
NASOPHARYNGITIS	0	6	0
NAUSEA	0	6	0
PRURITUS	8	26	23
RASH	5	11	13
SINUS BRADYCARDIA	0	8	7
UPPER RESPIRATORY TRACT INFECTION	6	0	0
VOMITING	0	7	0

Table 1. A table with a simplified adverse events summary

To provide more flexibility to customize table appearance, the r2rtf package provides four optional functions:

- `rtf_title()`: adds customized title and subtitle to the top of the table body.
- `rtf_colheader()`: customizes the table column header.
- `rtf_footnote()`: adds footnote to the bottom of the table body.
- `rtf_source()`: adds a data source to the bottom of the table body.

In particular, function `rtf_colheader()` shares parameter values with `rtf_body()` to control the format of the table column headers. Similarly, `rtf_title()`, `rtf_footnote()` and `rtf_source()` also share parameters with `rtf_body()` to control the text size, format, text alignment, etc. in the table title, footnote, and data source.

The r2rtf package stands out in the in-text management in the process of table generation. Our functions can handle special characters inside or outside the table body in UTF-8 encoding such as Greek, Symbol, Chinese, Japanese, Korean, etc.

Besides creating RTF tables, the r2rtf package is also designed to generate figures which can be exported in RTF format. We provide two unique functions that are required for figure generation. The function `rtf_read_png()` reads PNG figures into binary files, while `rtf_figure()` controls figure size and page width, height, and orientation. Another two required functions, `rtf_encode()` and `write_rtf()`, are shared between figure and table generation procedures. To add a title, footnote and data source to a figure, users can also use the functions `rtf_title()`, `rtf_footnote()`, and `rtf_source()` in the same way as in a table. In the definition code attached below, we use the minimal functions to read `filename` that contains the path to a figure in PNG format into R, encode it in RTF syntax, and write it into an RTF file:

```
filename %>%
  rtf_read_png() %>%
  rtf_figure() %>%
  rtf_encode(type = "figure") %>%
  write_rtf(file = "fig/fig-simple.rtf");
```

TABLE BODY

In the procedure to convert a data set into a table, the table body carries the most important information extracted from the data set. The name of the data set is the only required parameter in `rtf_body()`, and there are additional 30 optional parameters provided in the function that help users customize the table body appearance.

The complete set of 31 parameters in `rtf_body()` can be further classified into 6 groups.

Group 1 – Input R Data Sets

Users should first input an R data set `tbl` that needs to be converted to an RTF table, and then use `colheader` to indicate whether to include a default table header or not. The default setting is to add a default column header to the table body in case no customized column headers will be added to the table.

Parameter	Default Value	Explanation
`tbl`	`N/A`	A data frame
`colheader`	`TRUE`	A Boolean value to indicate whether to add default column header

Group 2 – Page Settings

The default page size setting is 8.5 inches in width, 11 inches in height for standard page size of a letter. Users can easily change parameter values in this group to adjust page width, height, orientation and doctype as they prefer. The doctype argument control the margins of a page. We provide 4 different margin size settings.

Parameter	Default Value	Explanation
`page_width`	`8.5`	Page width in inches
`page_height`	`11`	Page height in inches
`orientation`	`"portrait"`	Orientation in “portrait” or “landscape”
`doctype`	`"wma"`	Doctype in “wma”, “csr”, “wmm”, or “narrow”

Doctype	Left	Right	Top	Bottom	Header	Footer
`"wma"`	`1.25`	`1.0`	`1.5`	`1.0`	`0.5`	`0.5`
`"csr"`	`1.25`	`1`	`1.75`	`1.25`	`1.75`	`1.0`
`"wmm"`	`1.25`	`1.0`	`1.0`	`1.0`	`1.75`	`1.0`
`"narrow"`	`0.5`	`0.5`	`0.5`	`0.5`	`0.5`	`0.5`

Group 3 – Border Settings

Parameters in this group control the border type (e.g., single, double, dash, dot, etc.), border color (e.g., 657 different colors named in default R function `color()`), and the width of borderline in the table body. The vectorization of parameter inputs in this group is highly supported. The simple example in **Table 1** can demonstrate the default table border setting in the table body.

Parameter	Default Value	Explanation
border_left	"single"	Left border type
border_right	"single"	Right border type
border_top	NULL	Top border type
border_bottom	"double"	Bottom border type
border_color_left	"black"	Left border color
border_color_right	"black"	Right border color
border_color_top	"black"	Top border color
border_color_bottom	"black"	Bottom border color
border_width	15	Border width in twips (1 twips is 1/1440 inch)

Group 4 – Column and Cell Related Settings

We allow users to take full control of the total width of columns, relative width for each column according to the text length, cell height, and cell justification. The default total width of columns is designed to be the page width divided by 1.4. Each column by default has the same width but can be adjusted by a vector input that indicates the relative column width ratio. The default cell height is 0.15 inches, and the text contents are centered by default but can be adjusted in each cell.

Parameter	Default Value	Explanation
col_rel_width	1	Column relative width in a vector. Single value has the same width of all columns. e.g. c(2,1,1) refers to 2:1:1.
col_total_width	page_width/1.4	Column total width for the table
cell_height	0.15	Height for cell in twips (1 twip is 1/1440 inch)
cell_justification	"c"	Justification for cell

Group 5 – Text Settings

Text related parameters can be changed to control the text appearance in each cell of the table body such as text font, format, color, background color, justification, font size, and line space before and after the text. The simple example in **Table 1** can demonstrate the default text setting in the table cells.

Parameter	Default Value	Explanation
text_font	1	Text font type (1 is Times New Roman)
text_format	"single"	Text format
text_color	"black"	Text color
text_background_color	NULL	Text background color
text_justification	"c"	Justification for text
text_font_size	9	Text font size
text_space_before	15	Line space before text in twips (1 twip is 1/1440 inch)
text_space_after	15	Line space after text in twips (1 twip is 1/1440 inch)

Group 6 – Page Related Settings

Parameters in this group provide advanced features to the table appearance such as to break one table into separate pages, display a table in section grouping, and decide whether to append tables. Refer to the section of highlighted features, function calls, and table outputs for details.

Parameter	Default Value	Explanation
page_num	42	Number of rows in each page
page_by	NULL	Column names to group by sections in table
new_page	FALSE	A Boolean value to indicate whether to separate grouped table into pages by sections
last_row	TURE	A Boolean value to indicate whether the table contains the last row of the final table

TABLE COLUMN HEADERS

The r2rtf package allows users to add multiple levels of customized column headers to a table. To make this easy and flexible, we introduce the following function:

- `rtf_colheader()`: customizes the table column headers.

This function provides 2 required and 23 optional parameters to control the process. The name of the data frame and a string for the customized column header must be input to the function. The rest of 23 optional parameters are shared with function `rtf_body()` and are defined in the table body section. Shared parameters are defined with the same default values but can be changed to make further adjustments.

To add a column header to a table, `rtf_colheader()` first takes in the two required parameters: 1) a data set `tbl` as defined in the section of Group 1 – Input R Data Sets, and 2) a header string `colheader` in a special format that uses "|" to separate the input column names. When an empty name is needed for a column, insert a space between two vertical lines; e.g., "`name 1 | | name 3`". **Table 2** illustarates the customized table column headers. The code is provided to demonstrate how to input a column header string in the correct format. The resulting output table column headers are shown in the first two rows of **Table 2**.

The function `rtf_colheader()` has 1 page related parameter `page_width` defined in the section of Group 2 – Page Settings , 9 border related parameters defined in the section of Group 3 – Border Settings , and 4 column and cell related parameters defined in the section of Group 4 – Column and Cell Related Settings . Users can estimate the relative width of each column using argument `col_rel_width` to correctly align the borders of each cell in the table column headers with the table body. By default, the function will take the same column relative width ratio defined in the table body to produce the table column headers. The relative width is helpful to build flexible and complex column header. For example, in **Table 2**, the first column header has four columns with `col_rel_width = c(3, 4, 4, 9)` and the second column header and the table body has eight columns with `col_rel_width = c(3,1,3,1,3,1,3,5)`. The second and third columns of the second column header and table body are under the "baseline" column in the first column header, because the sum of the relative width of the second and third columns are the same as the relative width of first column. The function also provides 8 parameters defined in Group 5 – Text Settings to control text appearance in the table column headers. A new Boolean parameter `first_row` is introduced in this function to allow users to indicate whether the current customized column header should be inserted as the first row of the table.

To add more than one header with complex column arrangements to the table body, users can repeatedly call `rtf_colheader()`. In each function call, a column header string with column names separated by "|" is required. In the case of adding multiple column headers to a table, the parameter `first_row` identifies which is the first table column header and guarantees the table borderlines display correctly. After the first

table column header is added, the rest of the table column headers are then appended to the table in the order of repeated function calls to `rtf_colheader()`.

We show an example in **Table 2** attached below with two rows of complex and multi-level column headers. In **Table 2**, parameters `col_rel_width` and `first_row` are used in function `rtf_colheader()` to control the layout of the table column headers.

	Baseline		Week 20		Change from Baseline		
Treatment	N	Mean (SD)	N	Mean (SD)	N	Mean (SD)	LS Mean (95% CI)†
Study Drug	61	16.6 (4.41)	61	-6.6 (5.95)	61	-7.0 (9.16)	-7.0 (-8.58, -5.38)
Placebo	70	18.4 (6.34)	70	-9.0 (7.04)	70	-8.7 (8.54)	-8.7 (-10.17, -7.18)

Table 2. A summary table of an efficacy analysis

We provide the code below to convert the data set `tbl` to a table in RTF file in **Table 2**. The `tbl` data set is a summary table of an efficacy analysis on the change from baseline to week 20 between two treatment groups using the data prepared by the Drug Information Association scientific working group (DIASWG) (http://www.missingdata.org.uk/). It contains two rows of data, each row for one treatment group. First, we call function `rtf_colheader()` twice to add two column headers to the table. The output table column headers are in row 1 and row 2 of **Table 2**. When using this function, a string of column names separated by "|" is required in `colheader`. In this example, we include a special character, †, in the second table column header. Special characters in latex format (e.g. `\\dagger`) can be easily handled in the string text. We also use the parameter `col_rel_width` to adjust the relative column width for each table column header. The table column header in the first row has column relative widths of 3:4:4:9, while the second row has the same column relative widths as the table body. In the first call of `rtf_colheader()`, the parameter `first_row = TRUE` indicates this is the first row in the table. Second, we use the function `rtf_body()` to produce the table body with column relative widths of 3:1:3:1:3:1:3:5. Thus, table column headers can be aligned with columns in the table body in a logical way. The parameter `text_justification = c("l",rep("c",7))` sets the text to be left adjusted in the cells of the first column, and then center adjusted in the cells of the other 7 columns. Third, we call the function `rtf_encode()` to extract and render the table attributes to RTF encoding. Finally, the function `write_rtf()` saves the RTF encoding to an RTF file `table2.rtf`.

The code to produce **Table 2** is as follows based on the input data set `tbl`:

```
tbl %>%
  rtf_colheader(colheader = " | Baseline | Week 20 | Change from Baseline",
                col_rel_width = c(3, 4, 4, 9),
                first_row = TRUE) %>%
  rtf_colheader(colheader = "Treatment | N | Mean (SD) | N | Mean (SD) | N |
                 Mean (SD) | LS Mean (95% CI)\\dagger") %>%
  rtf_body(col_rel_width = c(3,1,3,1,3,1,3,5),
           text_justification = c("l",rep("c",7)) %>%
  rtf_encode() %>%
  write_rtf("table2.rtf");
```

TABLE TITLE, SUBTITLE, FOOTNOTE, AND DATA SOURCE

One challenge for the final report generation is that users may want to make changes to the text in the title, subtitle, footnote, or data source separately from the table body. In this case, to avoid edits to the code in `rtf_body()` that generates the table body, the r2rtf package provides three easy-to-use functions to customized the title, subtitle, footnote, and data source:

- `rtf_title()`: adds a customized title and subtitle to the top of the table body.
- `rtf_footnote()`: adds footnote to the bottom of the table body.
- `rtf_source()`: adds data source to the bottom of the table body.

The above three functions treat the title, subtitle, footnote, and data source as separate paragraphs outside the table body. Thus, users can easily take control of the contents inside and outside the table body.

In detail, `rtf_title()` provides 2 required and 15 optional parameters, and `rtf_footnote()` and `rtf_source()` each provide 2 required and 14 optional parameters. The name of the data set `tbl` and a string or a vector of strings are required in all three functions. For instance, `rtf_title()` provides the parameters `title` and `subtitle`, `rtf_footnote()` provides the parameter `footnote`, and `rtf_source()` provides parameter `source`. Each of these parameters (`title`, `subtitle`, `footnote`, and `source`) takes in a string or a vector of strings and each element in the string vector is printed as a separate line. In this way, users can simply call function `rtf_footnote()` once to add multiple footnotes to the table.

To discuss the parameters in detail, the above three functions have required parameters `title`, `footnote`, and `source`. Note that the parameter `subtitle` in function `rtf_title()` is optional. The remaining 14 parameters shared among the three functions can be further grouped into three sets. In set 1, there are 8 text related parameters defined in the section of **Error! Reference source not found.**Group 5 – Text Settings to control text font, format, color, background color, justification, font size, and line space before and after the text. In set 2, there are 4 paragraph related parameters – `space`, `indent_first`, `indent_left`, and `indent_right` that are used to adjust paragraph space and indent. In set 3, there are 2 page related parameters – `new_page` and `hyphenation` that are Boolean values used to indicate whether to start a new page and to use hyphenation.

The usage of functions `rtf_title()`, `rtf_footnote()`, and `rtf_source()` are further demonstrated in **Table 3** (see details in the section of table concatenation).

FIGURE GENERATION

The r2rtf package is also designed to generate a highly customized figures which can be exported to an RTF file. The procedure for figure generation is similar to table generation. To convert one or multiple PNG figures into RTF figures, the r2rtf package provides four required functions:

- `rtf_read_png()`: converts one or more PNG figures one rtf fileto binary files.
- `rtf_figure()`: customizes igure body.
- `rtf_encode()`: extracts and renders figure attributes to RTF encoding.
- `write_rtf()`: writes an RTF encoding string to an RTF document.

In detail, `rtf_read_png()` provides the only required parameter `file` that takes in a vector of PNG figure paths to combine all figures into one rtf file. Figures will be displayed in different pages after adding the title, footnote and data source if `rtf_title()`, `rtf_footnote()` and `rtf_source()` are used. `rtf_figure()` provides 1 required and 6 optional parameters listed below. The required parameter `tbl` is the name of the converted binary files generated by `rtf_read_png()`. The rest of the 6 optional parameters in `rtf_figure()` are used to adjust page and figure related appearance listed below:

Parameter	Default Value	Explanation
`tbl`	`NULL`	Converted binary files
`page_width`	`8.5`	Page width in inches
`page_height`	`13`	Page height in inches
`orientation`	`"portrait"`	Orientation in “portrait” or “landscape”
`doctype`	`"wma"`	Doctype in “wma”, “csr”, “wmm” or “narrow”
`fig_width`	`5`	The width of figures in inch
`fig_height`	`5`	The height of figures in inch

Many functions are shared in the table and figure generation process. For required functions, `rtf_encode()` and `write_rtf()` are shared except we set parameter `type = "figure"` in function `rtf_encode()` to produce figures. For optional functions, all three functions, `rtf_title()`, `rtf_footnote()`, and `rtf_source()` are shared to add title, subtitle, footnote, and data source to both table and figures.

In the following example, we show the code to convert a PNG figure to an RTF figure and save as `fig-simple.rtf`. We produce the figure using the four required functions with the minimal parameters, and the output sample RTF figure is in **Figure 1**.

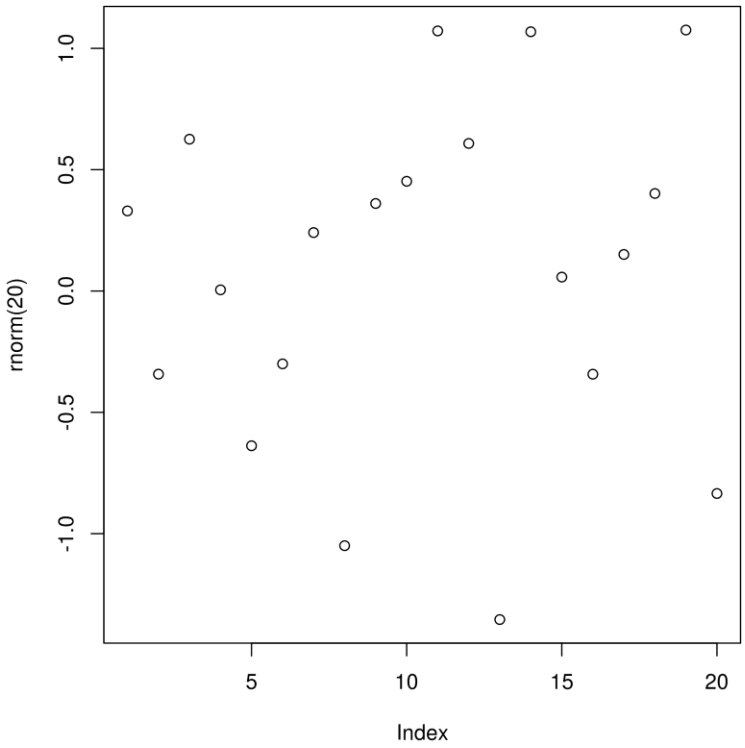

Figure 1. A sample figure output

The definition code to produce **Figure 1** is provided below:

```
filename <- "fig/fig1.png";
filename %>%
  rtf_read_png() %>%
  rtf_figure() %>%
  rtf_encode(type = "figure") %>%
  write_rtf(file = "fig/fig-simple.rtf");
```

HIGHLIGHTED FEATURES, FUNCTION CALLS, AND TABLE OUTPUTS

The r2rtf package provides more advanced features than simply converting an R data set to a table. In the following sections, we highlight several advanced features including appending multiple tables into one file, grouping tables by sections defined by the variable name in the data set, and table pagination under different conditions.

TABLE CONCATENATION

To minimize user efforts, we provide no extra functions for table concatenation. Users can simply use the existing functions to append tables by first producing a few tables by repeatedly calling `rtf_body()`. In each `rtf_body()` call, users should set the parameter value `last_row` that indicates whether the current

table is the last table that needs to be appended to the table list. Second, users need to put the generated tables in a list in the correct order. The order matters since our function concatenates the tables by the order in the list.

In the following example, we first take **Table 2** (see details in the section of table column headers) that was generated from table `tbl_1` and then append table `tbl_2` and `tbl_3` to the bottom of **Table 2**. Below we show `tbl_1`, `tbl_2` and `tbl_3` in **Figure 2, Figure 3**, and **Figure 4**.

Sample Data Set

	Trt	N1	Mean1	N2	Mean2	N3	Mean3	CI
1	Study Drug	61	16.6 (4.41)	61	-6.6 (5.95)	61	-7.0 (9.16)	-7.0 (-8.58, -5.38)
2	Placebo	70	18.4 (6.34)	70	-9.0 (7.04)	70	-8.7 (8.54)	-8.7 (-10.17, -7.18)

Figure 2. A summary of `tbl_1`

	comp	mean	p
1	Study Drug vs. Placebo	1.7 (-0.49, 3.88)	0.130

Figure 3. A summary of `tbl_2`

	rmse
1	Root Mean Squared Error of Change = 6.23

Figure 4. A summary of `tbl_3`

Table Output

We show the concatenation results in **Table 3** attached below.

ANCOVA of Change from Baseline at Week 8
Missing Data Approach
Analysis Population

	Baseline		Week 20		Change from Baseline		
Treatment	N	Mean (SD)	N	Mean (SD)	N	Mean (SD)	LS Mean (95% CI)†
Study Drug	61	16.6 (4.41)	61	-6.6 (5.95)	61	-7.0 (9.16)	-7.0 (-8.58, -5.38)
Placebo	70	18.4 (6.34)	70	-9.0 (7.04)	70	-8.7 (8.54)	-8.7 (-10.17, -7.18)
Pairwise Comparison				Difference in LS Mean (95% CI)†			p-Value
Study Drug vs. Placebo				1.7 (-0.49, 3.88)			0.130
Root Mean Squared Error of Change = 6.23							

†Based on an ANCOVA model.
ANCOVA = Analysis of Covariance, CI = Confidence Interval, LS = Least Squares, SD = Standard Deviation
Source:[study999: adam-adeff]

Table 3. A concatenated table of combined summary of an efficacy analysis

Below we provide the example code for table concatenation to produce **Table 3** with the title, subtitle, footnote, and data source defined in the default setting:

```
# convert tbl_1 to the table body. Add title, subtitle, two table
# headers, and footnotes to the table body.
tbl_1 %>%
  rtf_title(title = "ANCOVA of Change from Baseline at Week 8",
            subtitle = c("Missing Data Approach",
                         "Analysis Population")) %>%
  rtf_colheader(colheader = " | Baseline | Week 20 | Change from Baseline",
                col_rel_width = c(3, 4, 4, 9),
                first_row = TRUE) %>%
  rtf_colheader(colheader = "Treatment | N | Mean (SD) | N | Mean (SD) | N |
                 Mean (SD) | LS Mean (95% CI)\\dagger") %>%
  rtf_body(col_rel_width = c(3,1,3,1,3,1,3,5),
           text_justification = c("l",rep("c",7)),
           last_row = FALSE) %>%
  rtf_footnote(footnote = "\\daggerBased on an ANCOVA model.
                           \nANCOVA = Analysis of Covariance,
                            CI = Confidence Interval,
                            LS = Least Squares, SD = Standard Deviation");
# convert tbl_2 to the table body. Add a table column header to table body.
 tbl_2 %>%
  rtf_colheader(colheader = "Pairwise Comparison |
                Difference in LS Mean(95% CI)\\dagger | p-Value",
                text_justification = c("l","c","c")) %>%
  rtf_body(col_rel_width = c(8,7,5),
           text_justification = c("l","c","c"),
           last_row = FALSE);
# convert tbl_3 to the table body. Add data source to the table body.
 tbl_3 %>%
  rtf_body(colheader = FALSE,
           text_justification = "l") %>%
  rtf_source(source = "Source: [study999:adam-adeff]");
 # add tbl_1, tbl_1, and tbl_3 into a list in order
 tbl <- list(tbl_1, tbl_2, tbl_3);
 # concatenate a list of table and save to an RTF file
 tbl %>% rtf_encode() %>% write_rtf("table3.rtf");
```

SECTION GROUPING

Another useful feature we provide in our function `rtf_body()` is to produce a table with grouped sections through the parameter value `page_by`. Users can select a variable name in the R data set and input in parameter `page_by`. Then our function automatically groups the data into sections by the levels of the input variable, and a section header will be inserted at the beginning of each section. The section header contains a section name that is extracted from the levels of the selected variable. This advanced feature is further illustrated in the following example. We convert `tbl` (see **Figure 5**) to a table in **Table 4** attached below where we group the data into sections by “Gender”, “Age (Years)”, and “Race” defined in the variable “var_label”.

Sample Data Set

	var	1	1_pct	2	2_pct	3	3_pct	9999	9999_pct	var_label
1	Female	53	10.4	50	9.8	40	7.9	143	28.1	Gender
2	Male	33	6.5	34	6.7	44	8.7	111	21.9	Gender
3	<65	14	2.8	8	1.6	11	2.2	33	6.5	Age (Years)
4	>80	30	5.9	29	5.7	18	3.5	77	15.2	Age (Years)
5	65-80	42	8.3	47	9.3	55	10.8	144	28.3	Age (Years)
6										Age (Years)
7	Subjects with data	86		84		84		254		Age (Years)
8	Mean	75.2		75.7		74.4		75.1		Age (Years)
9	SD	8.6		8.3		7.9		8.2		Age (Years)
10	Median	76.0		77.5		76.0		77.0		Age (Years)
11	Range	52 to 89		51 to 88		56 to 88		51 to 89		Age (Years)
12	Black	8	1.6	6	1.2	9	1.8	23	4.5	Race
13	Caucasian	75	14.8	72	14.2	71	14.0	218	42.9	Race
14	Hispanic	3	0.6	6	1.2	3	0.6	12	2.4	Race
15	Other	0	0.0	0	0.0	1	0.2	1	0.2	Race

Figure 5. A sample data set on demographic and anthropometric characteristics

Table Output

We use the sample data set showed in **Figure 5** to produce an RTF table with grouped sections defined by the variable “var_label”. Three sections are displayed in the table with section headers – “Gender”, “Age (Years)”, and “Race”.

We show the definition code below to generate **Table 4**. Through the pipeline, we emphasize that the functions are easy-to-use and flexible to customize the table appearance.

```
tbl %>%
  rtf_title("Demographic and Anthropometric Characteristics",
            "ITT Subjects") %>%
  rtf_colheader(" | Placebo | Drug Low Dose | Drug High Dose | Total",
                col_rel_width = c(3, rep(2,4)),
                first_row = TRUE) %>%


  rtf_colheader(" | n | (%) | n | (%) | n | (%) | n |(%)",
                border_top = c("", rep("single", 8)),
                border_left = c("single", rep(c("single",""), 4)))) %>%
  rtf_body(page_by = "var_label",
           col_rel_width = c(3, rep(c(1.2, 0.8), 4)) ,
           text_justification = c("l", rep("d",8)),
           border_left = c("single", rep(c("single",""), 4) )) %>%
  rtf_footnote("This is a footnote", justification = "l") %>%
  rtf_source("Source: xxx", justification = "l") %>%
  rtf_encode() %>%
  write_rtf("table4.rtf");
```

Demographic and Anthropometric Characteristics
ITT Subjects

	Placebo		Drug Low Dose		Drug High Dose		Total	
	n	(%)	n	(%)	n	(%)	n	(%)
Gender								
Female	53	10.4	50	9.8	40	7.9	143	28.1
Male	33	6.5	34	6.7	44	8.7	111	21.9
Age (Years)								
<65	14	2.8	8	1.6	11	2.2	33	6.5
>80	30	5.9	29	5.7	18	3.5	77	15.2
65-80	42	8.3	47	9.3	55	10.8	144	28.3
Subjects with data	86		84		84		254	
Mean	75.2		75.7		74.4		75.1	
SD	8.6		8.3		7.9		8.2	
Median	76.0		77.5		76.0		77.0	
Range	52 to 89		51 to 88		56 to 88		51 to 89	
Race								
Black	8	1.6	6	1.2	9	1.8	23	4.5
Caucasian	75	14.8	72	14.2	71	14.0	218	42.9
Hispanic	3	0.6	6	1.2	3	0.6	12	2.4
Other	0	0.0	0	0.0	1	0.2	1	0.2

This is a footnote
Source: xxx

Table 4. A summary table of demographic and anthropometric characteristics

PAGINATION

There are two parameters in function `rtf_body()` – `page_num` and `new_page` that provide two approaches to control table pagination. The two parameters can either work separately or jointly to control pagination under different circumstances.

In the default page setting, we design each page to print a table with a maximum of 42 rows in portrait orientation, or 26 rows in landscape orientation. If the user produces a table with more than the maximum allowed rows in one page, our function will automatically break the table into multiple pages. If the user prefers to show fewer rows in a table for each page than the default value, we also provide a parameter `page_num`. The user only needs to provide an integer value to the parameter `page_num`, and the function will handle the desired pagination in the appropriate layouts.

Another pagination parameter we provide is `new_page`. A combination of section grouping and pagination is also supported in our function. In this case, `page_by` (introduced in the section of section grouping) and `new_page` can be used together to first group a table into sections and then print each grouped section as a full table in separate pages. At the same time, parameter `page_num` can work together with `new_page` to control the maximum rows allowed in a table in each page.

To simplify the process of pagination, there is no need to repeatedly call the same functions to control table appearance on each page. Customized features including the title, subtitle, table column headers, table body, footnote, and data source are carried out through every page automatically by one set of function calls. Only two parameters, `page_by` and `new_page`, are needed in function `rtf_body()` for pagination.

For example, **Table 4** (see the section of section grouping) can be further divided into three smaller tables in three pages if we set parameter `new_page = TURE` in function `rtf_body()`. On each page, it

displays the title, subtitle, table column header, table body with a section header and rows in one grouped section, footnote, and data source.

CONCLUSION

The r2rtf package provides a standard approach to produce fully featured and highly customized tables, listings, and figures in RTF format in the R platform. While providing powerful and flexible controls of table and figure appearance, all functions in the package are also easy-to-use and aim to minimize users' inputs. Therefore, R users at any skill level can easily use the r2rtf package to create desirable tables and figures either with simple features or with advanced features for complex layouts in regulatory environment.

We also carefully reduce the dependency of R packages to simplify the qualification process for regulatory use. This can make it extremely useful for a regulatory environment where well-tested software with the ability to produce this commonly needed output format is critical. The package enables both table and figure output building on tools that have previously been implemented in SAS® and have been honed through the test of time. The limitations of the package include that it is strictly oriented to the output in RTF file. One direction is to enhance the data structure used in the R package. For example, a data structure translation between r2rtf and gt R package maybe worthwhile to enable the table and figure customize features defined in r2rtf package can be easily exported in other commonly used format including HTML and PDF.

REFERENCES

Cunningham G. 1998. "An Easy-to-Use Macro for Use with Microsoft Windows That Converts Existing Text Tables to Microsoft Word Tables." Proceedings of Pharma*SUG 1998*.

Iannone R., Cheng J., Schloerke B. 2020. "gt: Easily Create Presentation-Ready Display Tables." R package version 0.1.0. Available at http://github.com/rstudio/gt.

Peszek I., Peszek R. 1998. "Automate the Creation and Manipulation of Word Processor Ready SAS Output." SAS Observations.

Peszek I., Song C., Kuznetsova O. 1999. "Producing Tabular Reports in SAS in the Form of Word Tables." Proceedings of Pharma*SUG 1999*.

Qi E., Zhang L. 2003. "%RTFTable – A Powerful SAS® Tool to Produce Rich Text Format Tables." Proceedings of Pharma*SUG 2003*.

Schaffer M. 2019. "rtf: Rich Text Format (RTF) Output." R package version 0.4-14. Available at http://CRAN.R-project.org/package=rtf.

Wehr p. 1996. "%PRINT Drivers: Teaching SAS to Speak the Many Languages of Document Publication." Proceedings of the 21st Annual SAS Users Group International Conference.

Wickham H., Francois R., Henry L., and Muller K. 2019. "dplyr: A Grammar of Dara Manipulation." R package version 0.8.3. Available at http://CRAN.R-project.org/package=dplyr.

Wickham H. 2019. "stringr: Simple, Consistent Wrappers for Common String Operations." R package version 1.4.0. Available at http://CRAN.R-project.org/package=stringr.

Zhang L., Qi E., Xu X. 2002. "%Rtfsymbol – An Easy Solution for Defining Special Symbols in RTF File – Generating SAS® Programs." NESUG 15 – 2002 proceedings.

Zhou J. 2001. "From SAS® ASCII File to Word Document – A SAS Macro Approach." Proceedings of Pharma*SUG 2001*.

ACKNOWLEDGMENTS

We would like to thank Eric Qi for discussions on the current and future advanced features of SAS® macros %RTFTable (Qi *el al.* 2003) and %Rtfsymbol (Zhang *el al.* 2002), which inspired us in the development of the r2rtf R package.

CONTACT INFORMATION

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

Siruo Wang
Johns Hopkins Bloomberg School of Public Health
sara.siruo@gmail.com

Simiao Ye
Merck & Co., Inc., Kenilworth, NJ, USA
simiao.ye1@merck.com

Keaven Anderson
Merck & Co., Inc., Kenilworth, NJ, USA
keaven_anderson@merck.com

Yilong Zhang
Merck & Co., Inc., Kenilworth, NJ, USA
yilong.zhang@merck.com