wfr: A Workflow to Assist the Creation of Documents Containing Many Tables and Figures

Jike Cui, Wenfei Zhang, Weiliang Qiu, Jun Luo, Liwei Wang, Donghui Zhang

2019-11-07

Table of Contents

1 Acknowledgment

The development of this workflow is inspired by Dr. Wenfei Zhang's previous work in separating data analysis from document creation - saving analysis results in R script and including those results in a R markdown file for document generation.

This work was supervised by Dr. Donghui Zhang at Sanofi.

Thanks to the authors of those excellent R package flextable, kableExtra, knitr, and bookdown.

2 Who can benefit from this package

The primary purpose of this package is to help saving time for users of the following tasks:

- Using R to carry out large data analysis project with many outputs, e.g. tables and figures, to save and manage.
- Creating post-analysis documents containing many tables and figures from the first step.

In addition, it ensures consistent formatting of the tables and numbers in all such documents created by different people in an organization.

3 Background and Objective

3.1 Non-statistical Work in Large Data Analysis Project

Significant amount of effort spent in a data analysis task is non-statistical.

A large data analysis project can produce many outputs, e.g. tables and figures, which are reviewed later. Annotating and managing all outputs are time-consuming but very important for future query, review, and understanding.

More importantly, in compiling reports for such projects, many of those output tables and figures are included. Table 1 is one such example. If a report contains two dozen such tables, compiling, formatting, and cross-referencing those tables take lots of time.

Table 1: Age and Sex statistics

variable	stats	subject[a]		
		case	control	all
Age (years)[b]	Number	14	12	26
	Mean (SD)	21 (9.7)	18.5 (9)	19.8 (9.3)
	Median	21.7	18.7	21.4
	Min : Max	4.3 : 33.4	4.6 : 34.4	4.3 : 34.4
Sex [n (%)]	Number	14	13	27
	Female	6 (42.9%)	5 (38.5%)	11 (40.7%)
	Male	8 (57.1%)	8 (61.5%)	16 (59.3%)

Note:

Table 1: Age and Sex statistics

variable	stats	subject[a]		
		case	control	all

[a]Caucasian[ref_1] only

[b]One control has no record

3.2 How to reduce the amount of non-statistical work

So the objective of this package is to help to reduce the portion of non-statistical work in this process; specifically:

- It provides a way to systematically record the output from the data analysis R script, even including format information if the output is a table.
- It can automatically create the Rmd file to produce either docx, html, or pdf output.
- Numeric columns of the tables in the report are formatted automatically and properly based on the distribution of the column.

4 The Workflow

Figure 1 includes four pictures depicting the proposed workflow in conducting data analysis project:

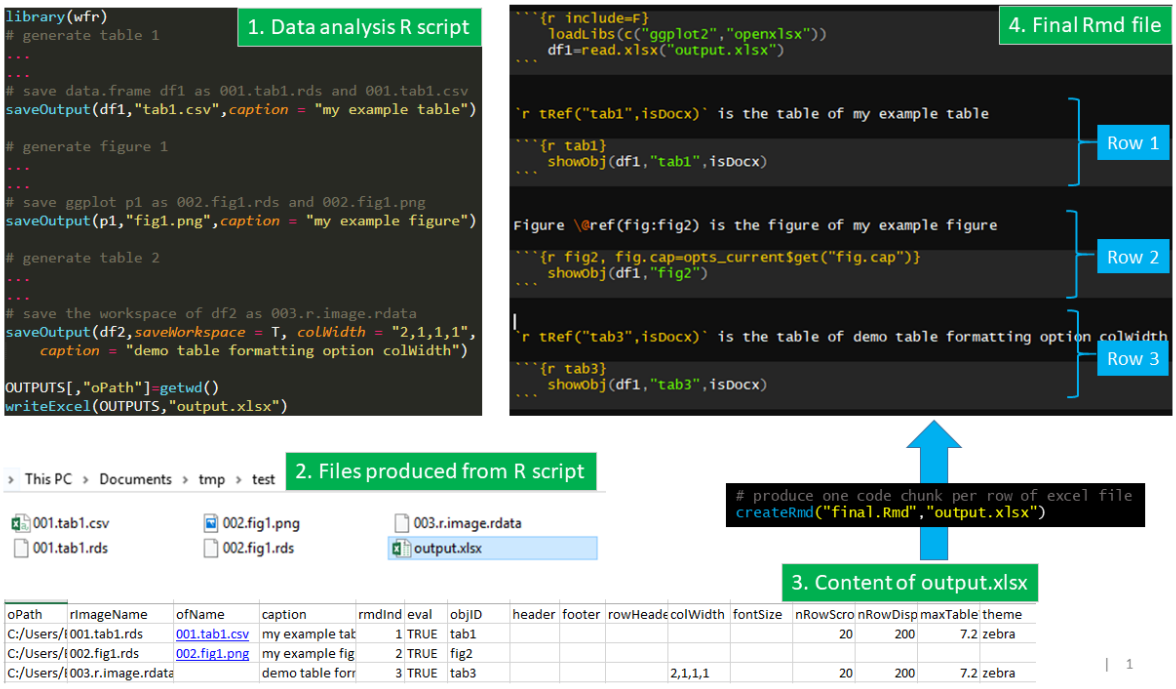

Figure 1: Proposed workflow from R script to Rmd file

1. In the data analysis R script, use `saveOutput()` to save each object you want to keep with a caption to explain what the object is. `saveOutput` saves either the current R workspace or the rds file of the object, depending on the flag `saveWorkspace`. A text or picture file, e.g. `csv` or `png`, of the object can also be saved via `oFileName`. A number of formating options, if the object is a table, can also be saved through corresponding parameters of `saveOutput`.

   `saveOutput` also enters all those information into a global data.frame, `OUTPUTS`, which is saved as an excel file at the end of R script.

2. The file explorer shows all the saved files and the excel file.

3. The content of the excel file can be manually edited. The `oFileName` cells are imbedded with links to the file, and clicking opens the files directly.

4. Run function `createRmd()` to produce the Rmd file for the report in either html or docx format. `wfr` package provides a R markdown template file that can output docx, html, or pdf format depending by setting `oFormat` in the file. Appending to this Rmd template file, `createRmd` firstly adds a code chunk to read in the excel file, then one code chunk per row of the excel file for cross reference and display of the object in the row. If the object is a table, its numerical columns are automatically formatted based on the min, max, and median values of the column by num2formattedStr. Also see showObj for details on how it displays an object.

Based on this auto-generated Rmd file, users can focus on the text part of the report, saving significantly amount of time. A few lines of the Rmd file needs to be customized before knitting, i.e. `oFormat, isDocx, outputFileName, title, author`.

5 Specific Functions

5.1 showObj

It displays an object in knitr code chunk and set the caption with cross reference. Figure 2 depicts how it works:

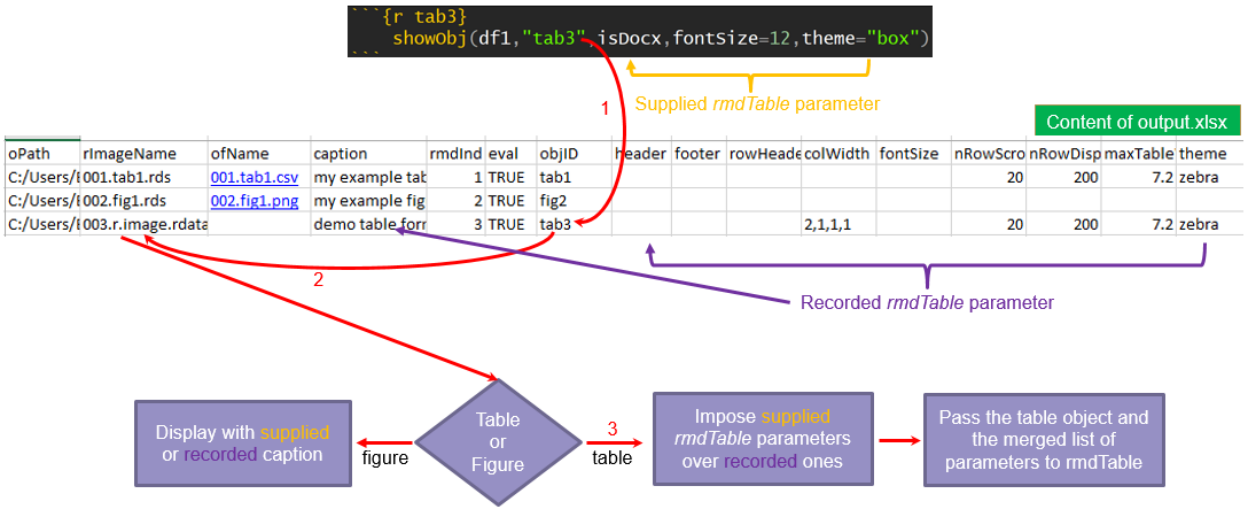

Figure 2: Under the hood of `showObj`

1. Identifies the row containing the `objID` in the data.frame read in from the excel file.
2. Reads the object under `rImageFile` in that row.
3. Display the object if it's a figure, or pass it to `rmdTable` if a table, along with formatting info either passed in via `showObj` or read from other columns in that row of the data.frame.

Also note that the chunk labels are passed to `showObj` as `objID` whose default value is actually `NULL`, meaning it is not a required parameter. In knitting the Rmd file, `showObj` can grab the chunk label automatically if `objID` is not supplied. But by supplying `objID`, a chunk can be run alone, e.g. for debugging purpose if the display from `showObj` is wrong. To debug, just type `debugonce(showObj)` in R console, and read in necessary variables, e.g. `df1` and `isDocx`, and run the chunk.

5.2 rmdTable

5.2.1 How to build a table

This is a wrapper function to plot tables using either `flextable` or `kable` depending on a flag `isDocx`. If `isDocx` is `TRUE`, the default, `flextable` is used because `kable` cannot produce complex tables for Word document. Below are some examples on how to use this function.

Figure 3 shows the original table, the final table we want to create, and the intermediate tables; note the name of different sections of a table, esp. *row header*, the columns in the table body used as the header of rows.

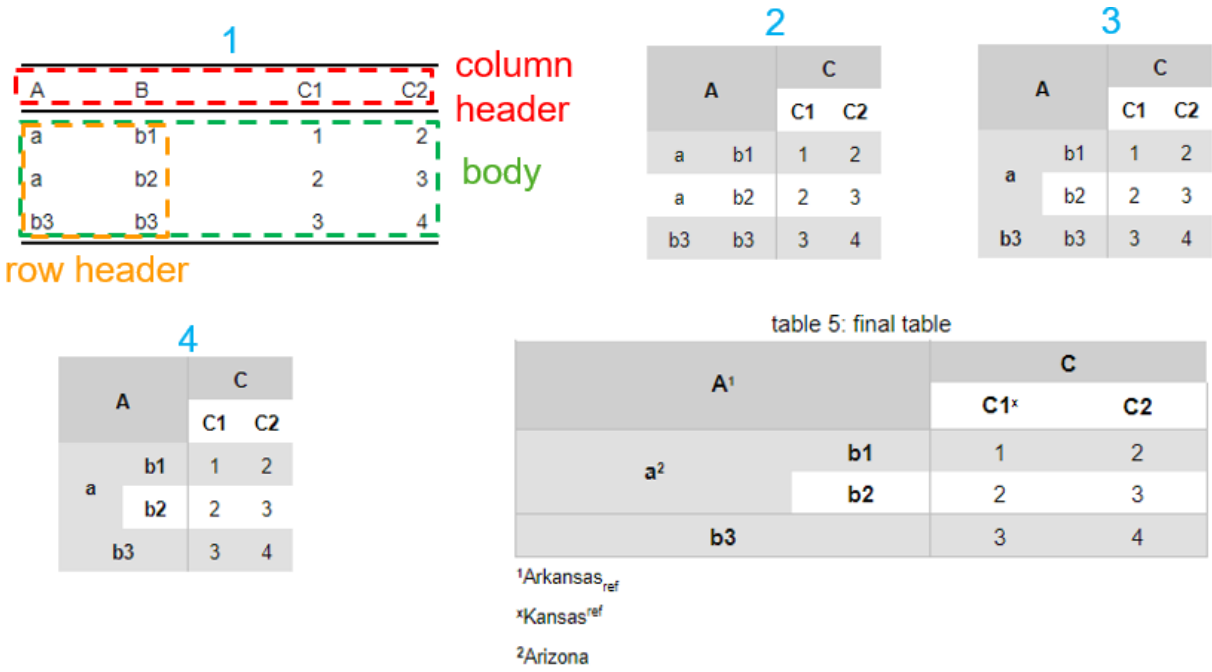

Figure 3: From the original to the final table

The below code chunk shows how each of the table in Figure 3 are created, `df1` contains the content forming the original table. The header of `df1` has only one row, whereas the header of the final table have two rows. Therefore a new header needs to be created with two vectors, corresponding to the two rows in the final table. `rmdTable` **automatically merges neighboring cells in the header if their contents are identical**, e.g. the four `A` cells, and the two `C` cells as illustrated in intermediate table 2.

```
df1=data.frame(A=c("a","a","b3"), B=c("b1","b2","b3"), C1=1:3, C2=2:4)
flextable(df1) #table 1

columnHeader=list(c('A','A','C','C'), c('A','A','C1','C2'))
rmdTable(df1,header = columnHeader) #2

rmdTable(df1,header = header, rowHeaderInd = 1) #3

rmdTable(df1,header = header, rowHeaderInd = 2) #4

footer=list(c("A","Arkansas$~ref$",'1','header'),
            c("C1","Kansas$^ref$",'x','header'),
            c('a',"Arizona",'2','body'))

rmdTable(df1,header = header,
         rowHeaderInd = 2,
         footer = footer,
         caption = "table 5: final table",
         colWidth = c(2,1,1,1),
         fontSize = 12) #5
```

```
# setting as character string
colWidth = "2,1,1,1"
header = "A | A | C | C || A | A | C1 | C2"
footer = "A|Arkansas$~ref$|1|header ||
          C1|Kansas$^ref$|x|header   ||
          a|Arizona|2|body"
```

`rowHeaderInd` is an integer and means column `1:rowHeaderInd` in the table body are going to be the row header. Setting `rowHeaderInd` has two effects on the row headers, as seen in intermediate table 2 and 4:

- Just like in the column header, **neighboring cells of identical content are collapsed into one multi-row/column cell**.
- Font becomes bold.

There are a number of additions in the final table. Probably the most important one is footer; it's a list of character vector which contains *the content of cell to tag*, *the content of footer*, *the tag symbol*, *the section of the table to identify the cell*. Some parameters can also be set as string so that they can be recorded into the excel file in Figure 1, either manually or via `saveOutput`. See the documentation of `rmdTable` for a detailed explanation on `footer` and how to construct the strings.

5.2.2 Table theme

Figure 4 lists all the themes available for docx output, which is based on `flextable` but with the addition of the light grey line inside the table.

table theme: zebra

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

table theme: box

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

table theme: tron

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

table theme: booktabs

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

table theme: vanilla

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

table theme: vader

A[1]		C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3		32.0	3.25e-01

Figure 4: Table themes for Word output

For html output, function `kable` is used, and only zebra, box, and vanilla are available. In addition, I have not found a way, using `kable`, to merge horizontal neighboring cells in the

first row of header (the two `A`) and in the table *body* (the two `b3`), as figure 5 shows. Although such needs for more table themes and merging those cells are rare, if really needed, `flextable` can be used for html output by just setting `isDocx = TRUE` instead of the `isDocx = (oFormat != "html")` in the Rmd file produced by `createRmd`. The only drawback is that the cross-reference of flextable-produced tables is not URL-linked, i.e. not clickable.

table theme: box

A[1]	A[1]	C	
		C1[x]	C2
a[2]	b1	1,001.1	-3.21e-03
	b2	58.0	1.21e-02
b3	b3	32.0	3.25e-01

[1]A: Arkansas[ref]
[x]C1: Kansas[ref]
[2]a: Arizona

Figure 5: `kable` cannot merge horizontal cells in the 1st row of header and table body

5.3 tRef and tCap

When `flextable` is used, that is, when the output format of the Rmd file is Word, the general way to do cross reference for tables, i.e. *`Table \@ref(tab:label)`*, does not work. Therefore, `tRef` and `tCap` are created to handle table cross reference in Rmd text and table caption, regardless of the output format.

- **`` `r tRef("label")` ``** replaces **`Table \@ref(tab:label)`** in Rmd text
- To use `rmdTable`, parameter `caption` should be set as `tCap(original.caption,"label",isDocx)` to enable cross reference in caption, e.g. `caption = tCap("Age and Sex statistics","tab1",isDocx)` for Table 1.

5.4 num2formattedStr

Each numerical column of the `dataDf` in `rmdTable` is formatted automatically by function `num2formattedSt`. This function makes decision on the following four aspects of formatting based on the min, median, max values of a numeric vector, so that the

formatted numbers carry enough information and are in a length less than 10 characters. For example, for a numeric vector `v`, if `median(abs(v)) > 100`, it makes no sense to keep decimal digits.

- number of sinificant digits
- number of decimal points
- whether to apply 1000 separator ‘,’
- whether to apply scientific notation

To illustrate, Table 2 is before formatting, and Table 3 is after formatting.

```
fileName = system.file("extdata", "example.format.csv", package = "wfr")
df1 = read.csv(fileName, check.names = FALSE)
flextable(df1) %>%
set_caption(caption = tCap("Auto-formatting on numbers using flextable",
            "tabNoF", isDocx)) %>% autofit()
```

Table 2: Auto-formatting on numbers using flextable

variable	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1131.217	378760.729	0.003	0.998
chr12_6	-113.319	35949.291	-0.003	0.997
chr1_16	8.226	5506.544	0.001	0.999
chr2_31	-71.597	22311.468	-0.003	0.997
chr19_31	-77.541	24585.401	-0.003	0.997
chr5_31	80.417	26666.898	0.003	0.998
chr10_31	41.732	14357.733	0.003	0.998
chr1_6	-12.562	6032.184	-0.002	0.998

```
rmdTable(df1, caption = tCap("Auto-formatting on numbers using
         num2formattedStr", "tabF", isDocx), isDocx = isDocx)
```

Table 3: Auto-formatting on numbers using num2formattedStr

variable	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1,131.2	378,761	2.99e-03	0.998
chr12_6	-113.3	35,949	-3.15e-03	0.997
chr1_16	8.2	5,507	1.49e-03	0.999

Table 3: Auto-formatting on numbers using num2formattedStr

variable	Estimate	Std. Error	z value	Pr(>|z|)
chr2_31	-71.6	22,311	-3.21e-03	0.997
chr19_31	-77.5	24,585	-3.15e-03	0.997
chr5_31	80.4	26,667	3.02e-03	0.998
chr10_31	41.7	14,358	2.91e-03	0.998
chr1_6	-12.6	6,032	-2.08e-03	0.998

5.5 setWidths

For non-html output, this function properly and automatically sets the width of each column so that the width of the whole table does not exceed the maximum width set by `maxTableWidth`. The default value of `maxTableWidth` is 7.0 inches, which corresponds to "PAGE LAYOUT" > "size" > "letter" and "Margins" > "Moderate" in MS Word. 6.5 inches corresponds to "Margins" > "Normal".

The width of the whole table is the sum of its column. However, computing such width can be very tricky:

- The width of a table cell depends on the font family, font size, font type (bold, italic, etc), and table theme. For large and complex tables which barely fits the `maxTableWidth`, change of any those factors may mess up some columns. function `setWidths` is only tested for font Arial, size 12 - 7, normal and bold type, and theme zebra.
- The width of a column is the maximum width among all cells in that column, including header and table body. If the sum of column widths does not fit the `maxTableWidth`, there are a few options to reduce the column widths:
  - Wrapping the long headers
  - Reducing the font size down to `minFontSize`
  - Wrapping the long columns in table body

In majority cases, the column widths set by `setWidths` does not need any manual adjustment. This function is called by rmdTable internally. See manual for its algorithm to determine the width of each column. Following are a few illustrations.

Table 4 sets `maxTableWith = 7`. The default font size is 11. The algorithm chooses wrapping headers over reducing font size. Headers are wrapped in a way so that

- headers of single word are not wrapped, even if they are much longer than its table body;

- other headers whose lengths are greater than those of table body are all wrapped once, instead of some not being wrapped and some being wrapped twice. Such behavior is achieved by finding the non-letter character closest to the center of a header and using that as wrapping point. The length of the header is updated to be the length after wrapping after addition of a proper cell padding.

```
fileName = system.file("extdata", "example.table.2.csv", package = "wfr")
df1 = read.csv(fileName, check.names = FALSE)
colnames(df1) = gsub("?", "Δ", colnames(df1), fixed = TRUE)
header = list(c("pathway", "gene", rep("batch A", 3), rep("batch B", 3),
              rep("A vs B", 3)), colnames(df1))
footer = list(c("ΔCt Mean", "ΔCt is calulated using median value as reference
                gene for each visit and patient,i.e., ΔCt = Ct –
                Ct$~median$", "a", "header"),
                c("Gene 1", "Our target gene$^ref$", "b", "body"))
rmdTable(df1, caption = tCap("maxTableWidth is set as 7.0 inches (default)",
                             "tabSW1", isDocx),
         footer = footer, header = header, rowHeaderInd = 1, isDocx = isDocx)
```

Table 4: maxTableWidth is set as 7.0 inches (default)

pathway	gene	batch A			batch B			A vs B		
		N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
A	Gene 1[b]	40	3.73	0.194	15	3.12	0.319	0.808	1.75	0.049
	Gene 2	42	2.41	0.152	15	1.79	0.164	0.570	1.49	0.059
	Gene 3	37	0.39	0.150	14	-0.18	0.158	0.540	1.45	0.083
B	Gene 4	36	3.88	0.151	16	3.42	0.268	0.647	1.57	0.095
	Gene 5	40	-3.02	0.148	13	-1.86	0.722	-0.904	-1.87	0.098
	Gene 6	40	2.14	0.231	14	1.19	0.389	0.982	1.98	0.103

[a]ΔCt is calulated using median value as reference gene for each visit and patient,i.e., ΔCt = Ct - Ct_{median}
[b]Our target gene[ref]

Table 5 sets `maxTableWith = 6.5`. The algorithm reduces font size by 1 to fit the table. Note that the font size of the table title and footer is not affected. The former has a minimum size 10, and the latter is set by `footerFontSize` in rmdTable, which defaults at 9.

Table 5: maxTableWidth is set as 6.5 inches

pathway	gene	batch A			batch B			A vs B		
		N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
A	Gene 1[b]	40	3.73	0.194	15	3.12	0.319	0.808	1.75	0.049
	Gene 2	42	2.41	0.152	15	1.79	0.164	0.570	1.49	0.059

Table 5: maxTableWidth is set as 6.5 inches

pathway	gene	batch A			batch B			A vs B		
		N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
	Gene 3	37	0.39	0.150	14	-0.18	0.158	0.540	1.45	0.083
B	Gene 4	36	3.88	0.151	16	3.42	0.268	0.647	1.57	0.095
	Gene 5	40	-3.02	0.148	13	-1.86	0.722	-0.904	-1.87	0.098
	Gene 6	40	2.14	0.231	14	1.19	0.389	0.982	1.98	0.103

[a]ΔCt is calulated using median value as reference gene for each visit and patient,i.e., ΔCt = Ct - Ct_{median}
[b]Our target gene[ref]

Table 6 sets `maxTableWith = 6.5` and `minFontSize = 9`. Due to some long cells in table body, reducing the font size to 9 still does not fit 6.5 inches. The long columns of table body, i.e. *pathway* and *gene*, are wrapped in a way that

- columns in `rowHeaderInds`, e.g. *pathway*, may be wrapped more than once if each of its cells spans multiple cells.
- other columns are wrapped as little as possible to avoid unnecessary wrapping; for example, "*Gene 4 w/ long names*" is not wrapped.

```
fileName = system.file("extdata", "example.table.3.csv", package = "wfr")
df1 = read.csv(fileName, check.names = FALSE)
colnames(df1) = gsub("?", "Δ", colnames(df1), fixed = TRUE)
rmdTable(df1,
     caption = tCap("maxTableWidth is set as 6.5 inches ", "tabSW4", isDocx),
     maxTableWidth = 6.5, footer = footer, header = header,
     rowHeaderInd = 1, minFontSize = 9, isDocx = isDocx)
```

Table 6: maxTableWidth is set as 6.5 inches

pathway	gene	batch A			batch B			A vs B		
		N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
Insulin signal transduction pathway	Gene 1[b]	40	3.73	0.194	15	3.12	0.319	0.808	1.75	0.049
	Gene 2 has a long long name here	42	2.41	0.152	15	1.79	0.164	0.570	1.49	0.059
	Gene 3	37	0.39	0.150	14	-0.18	0.158	0.540	1.45	0.083
Apoptosis Through Death Receptors	Gene 4 w/ long names	36	3.88	0.151	16	3.42	0.268	0.647	1.57	0.095
	Gene 5	40	-3.02	0.148	13	-1.86	0.722	-0.904	-1.87	0.098
	Gene 6	40	2.14	0.231	14	1.19	0.389	0.982	1.98	0.103

[a]ΔCt is calulated using median value as reference gene for each visit and patient,i.e., ΔCt = Ct - Ct_{median}
[b]Our target gene[ref]

Table 7 does not reduce font size since the algorithm sees column *pathway* can be wrapped multiple times.

Table 7: maxTableWidth is set as 6.5 inches

pathway	batch A			batch B			A vs B		
	N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
Insulin signal transduction pathway	40	3.73	0.194	15	3.12	0.319	0.808	1.75	0.049
	42	2.41	0.152	15	1.79	0.164	0.570	1.49	0.059
	37	0.39	0.150	14	-0.18	0.158	0.540	1.45	0.083
Apoptosis Through Death Receptors	36	3.88	0.151	16	3.42	0.268	0.647	1.57	0.095
	40	-3.02	0.148	13	-1.86	0.722	-0.904	-1.87	0.098
	40	2.14	0.231	14	1.19	0.389	0.982	1.98	0.103

[a]ΔCt is calulated using median value as reference gene for each visit and patient,i.e., $\Delta Ct = Ct - Ct_{median}$

In contrary, in Table 8 the algorithm prefers reducing font size over wrapping column *gene* since its table body cells are all single row.

Table 8: maxTableWidth is set as 6.5 inches and minFontSize as 9

gene	batch A			batch B			A vs B		
	N	ΔCt Mean[a]	ΔCt SEM	N	ΔCt Mean	ΔCt SEM	LS mean	Fold change	p-value
Gene 1[b]	40	3.73	0.194	15	3.12	0.319	0.808	1.75	0.049
Gene 2 has a long long name here	42	2.41	0.152	15	1.79	0.164	0.570	1.49	0.059
Gene 3	37	0.39	0.150	14	-0.18	0.158	0.540	1.45	0.083
Gene 4 w/ long names	36	3.88	0.151	16	3.42	0.268	0.647	1.57	0.095
Gene 5	40	-3.02	0.148	13	-1.86	0.722	-0.904	-1.87	0.098
Gene 6	40	2.14	0.231	14	1.19	0.389	0.982	1.98	0.103

[a]ΔCt is calulated using median value as reference gene for each visit and patient,i.e., $\Delta Ct = Ct - Ct_{median}$
[b]Our target gene[ref]

6 Template Files

For Word output, the formatting style is based on a template file `system.file("extdata", "word.template.for.Rmd.dotx", package = "wfr")`. Reads its content to see how to know how its style corresponds to R markdown syntax and how to edit its style. See this article for more details.

7 Installation and testing the package

7.1 Dependency

flextable_0.5.5 is required.

For docx output from Rmd file, pandoc version >= 2.0 is needed. Rstudio is bundled with pandoc, but only version >= 1.2 has the needed pandoc version. So update your Rstudio if necessary.

For pdf output of Rmd file, a LaTeX engine is needed. If not available, run the following code and restart Rstudio after it finishes. However, this and most LaTeX engine do not accept unicode characters; it might be easier to produce MS Word document and then export it as pdf file.

```
install.packages(c('tinytex', 'webshot'))
tinytex::install_tinytex()
webshot::install_phantomjs()
```

7.2 Testing

Run the following code for a simple testing.

```
install.packages(c("flextable","knitr","kableExtra","captioner","officer",
                   "openxlsx","bookdown", "pander","devtools"),
                 dependencies=TRUE)

# installation from source; add path to the file
devtools::install_github("blueskypie/wfr",dependencies=TRUE)

# testing
library(wfr)
library(ggplot2)

OFCOUNTER=1
df1=data.frame(A=c("a","a","b3"),
               B=c("b1","b2","b3"),
               C1=c(1001.123,58.04,32.01),
               C2=c(-0.00321, 0.0121, 0.325))

# output directory; set your own directory if preferred.
outDir = tempdir()
ofn = file.path(outDir, "tab.1.csv")
saveOutput(df1, oFileName=ofn,
           caption="testing table w/o formatting info")

ofn = file.path(outDir, "fig.1.png")
```

```
saveOutput(qplot(1:10,1:10), oFileName=ofn,
           caption="this is a testing plot")

colWidth = "2,1,1,1"
header = "A | A | C | C || A | A | C1 | C2"
footer = "A|Arkansas$~ref$|1|header
          || C1|Kansas$^ref$|x|header
         || a|Arizona|2|body"
rowHeaderInd = 2

ofn = file.path(outDir, "tab.2.csv")
saveOutput(df1, oFileName = ofn,
           caption = "testing table w/ formatting info",
           header = header, footer = footer, colWidth = colWidth,
           rowHeaderInd = rowHeaderInd)

wfrFile=system.file("extdata", "example.table.2.csv", package = "wfr")
df1 = read.csv(wfrFile,check.names = FALSE)
header="pathway | gene | batch A | batch A | batch A | batch B |
        batch B | batch B | A vs B | A vs B | A vs B ||
        pathway | gene | N | ΔCt Mean | ΔCt SEM | N |
        ΔCt Mean | ΔCt SEM | LS mean | Fold change | p-value"
footer="ΔCt Mean | ΔCt is calulated using median value as reference gene for
        each visit and patient,i.e., ΔCt = Ct - Ct$~median$ | a | header"

ofn = file.path(outDir, "tab.3.csv")
saveOutput(df1, oFileName ="tab.3.csv",
           caption = "testing a complex table",
           header = header, footer = footer,
           rowHeaderInd = 1, maxTableWidth = 6.5)

exFn = file.path(outDir, "all.outputs.xlsx")
# open the file to see its content.
writeExcel(exFn)

# Open the created Rmd files, change the output format "oFormat" to see if
# all output formats work. Add a few headings so Table of Content can be
# created and also see the effect of the docx reference template for docx
# output.
rmdFn1 = file.path(outDir, "all.outputs1.Rmd")
createRmd(rmdFn1,exFn)

rmdFn2 = file.path(outDir, "all.outputs2.Rmd")
# add additional formatting to all tables
createRmd(rmdFn2,exFn, tabPars = "fontSize=12,theme = 'booktabs'")
```