PharmaSUG 2022 - Paper QT-263

R Tables via GT for Regulatory Submissions

Phil Bowsher, Posit/RStudio PBC;
Rich Iannone, Posit/RStudio PBC

Abstract

The gt package helps statistical programmers create wonderful-looking tables using the R programming language. TLGs (tables, listings, and graphs - also known as TLFs/TFLs with F for figures) are an important part of clinical trial reporting. The goal of this paper is to introduce statistical programmers to the gt package for table creation in submissions to regulatory bodies. This paper is aimed at statistical programmers that are building clinical tables and have wondered how to approach TLGs with the gt R package. The gt package is part of the Guidance Document for the use of affiliated R packages in Regulated Clinical Trial Environments here:

https://resources.rstudio.com/assets/img/validation-tidy.pdf

A keynote presentation about using gt in pharma can be found here:

https://rinpharma.com/publication/rinpharma_202/

The main gt site for reference is here:

https://gt.rstudio.com/

You can test out gt live here via Posit Cloud via:

https://rstudio.cloud/project/779965

The examples below are maintained at the following github link:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt

Introduction

The gt package is a table package fully supported by Posit/RStudio PBC, where work started in early 2018. Its first release to CRAN was in April of 2020. The primary purpose of the package is to help users easily generate information-rich, publication-quality tables from R. Like ggplot2 (deriving from Grammar of Graphics approach to layering aesthetic elements), the name gt is short for “grammar of tables”. gt is based on information from the Manual of Tabular Presentation by the Bureau of the Census. The document is focused on the theory and practice in the presentation of statistical data in tables for publication.

https://www2.census.gov/library/publications/1949/general/tabular-presentation.pdf

Both ggplot2 and gt describe a set of underlying components that can be recombined in different ways to solve different problems. This grammar helps developers to enforce good and sensible data-management practices when creating tables.

Pharmaceutical companies often have unique needs around TLGs. Some will use RTF while others prefer pdfs or Word documents. Often unique formatting requirements also exist for items such as complex headers or multi-page formats. To help meet the evolving needs of pharmaceutical requirements, the gt package also changes to meet these specifications. Various pharmaceutical requests were added to gt that came from feedback from pharmaceutical statistical programmers. Much of the updates are documented here:

https://posit.co/wp-content/themes/Posit/public/markdown-blogs/changes-for-the-better-in-gt-0-6-0/index.html#better-rtf-output-tables-for-pharma

If you discover that features are missing, please create an “issue” on github here:

https://github.com/rstudio/gt/issues

You can see an example from R in Pharma below:

https://github.com/rstudio/gt/issues/653

In R, there is a burgeoning collection of packages for display table generation. Many are listed here:

https://gt.rstudio.com/#how-gt-fits-in-with-other-packages-that-generate-display-tables

https://pharmaverse.org/nonpharma/

Pharmaceutical companies have also created packages that work on top of gt such as:

https://github.com/GSK-Biostatistics/tfrmt

Another example is the {gto} package. Its function body_add_gt() allows gt tables to be inserted directly into Word documents using the standard {officer} workflow. For example, clinical tables can be made with tfrmt mentioned above, then Analysis Results Dataset can be rendered and then inserted into reports.

https://github.com/GSK-Biostatistics/gto

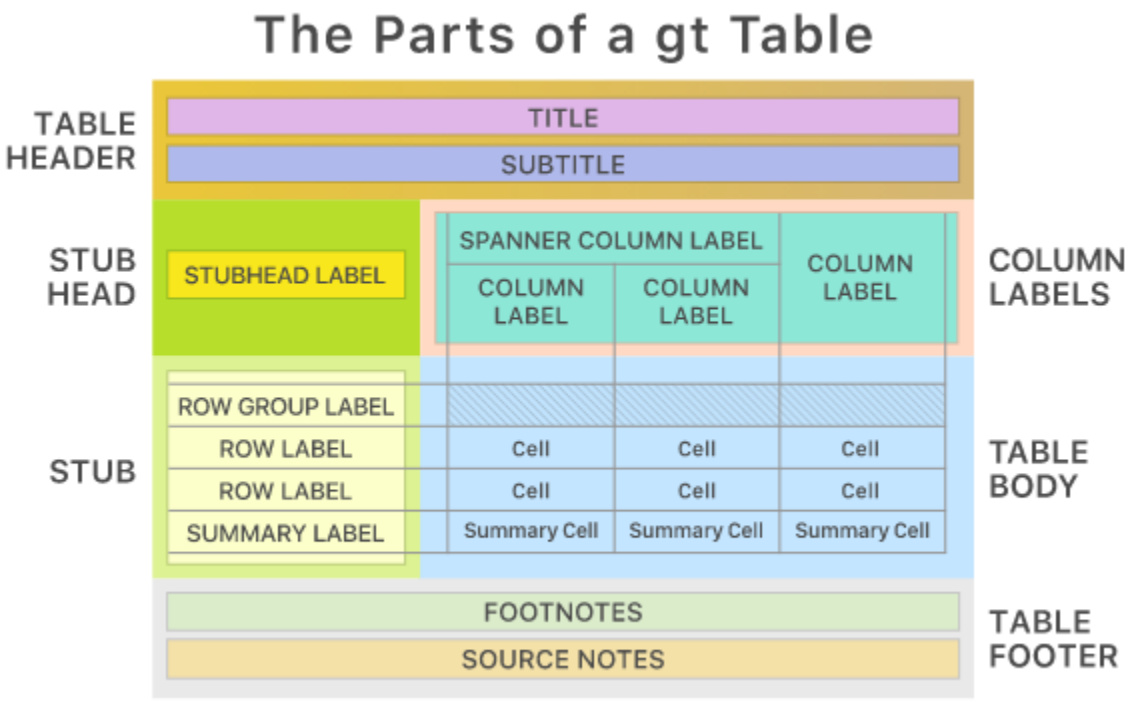
Other packages that utilize gt include gtextras and gtsummary.

https://jthomasmock.github.io/gtExtras/

https://www.danieldsjoberg.com/gtsummary/

Understanding GT

gt is best understood by familiarizing yourself with the parts of a table as shown below:



The main parts are table header, the stub, the column labels and spanner column labels, the table body, and the table footer. Title and footnotes are integral parts of Tables, Figures or Listings (TFLs) in Clinical Study Reports (CSRs) and Case Report Tabulation (CRT). Titles and footnotes are created or typed in a requirement document such as Statistical Analyses Plan (SAP). Please refer to the E3 Structure and Content of Clinical Study Reports:

https://www.fda.gov/regulatory-information/search-fda-guidance-documents/e3-structure-and-content-clinical-study-reports

Next it is important to understand the data to be displayed. The table data set is first converted to a gt object (with the gt() function). The user then uses the other functions from the gt package to format cells, add parts to the table (e.g., header and footer), modify the arrangement of the rows and columns, and format the data cells. The package allows for many different output formats like HTML, LaTeX/PDF, RTF, and Word. Any gt table can be exported manually to these formats (with the gtsave() function). Furthermore, gt tables can be integrated in R Markdown and Quarto documents, where the tables will be native to the output format for the entire document.

There are a lot of educational resources for helping the first-time user to get started. On the main website (at gt.rstudio.com) there are several introductory articles, including a quick-start article. There is a reference section that explains each of the package's functions, with copious easy-to-understand examples included along with images of output. There is a public Posit Cloud project available that allows you to experiment with hundreds of pre-built gt examples in a series of R Markdown documents.

Below is an example of gt:

```
library(gt)

islands_tbl <-
  dplyr::tibble(
    name = names(islands),
    size = islands
  ) %>%
  dplyr::arrange(desc(size)) %>%
  dplyr::slice(1:10)

gt_tbl <- gt(data = islands_tbl) %>%
  tab_header(
    title = "Large Landmasses of the World",
    subtitle = "The top ten largest are presented"
  ) %>%
  tab_source_note(
    source_note = "Source: The World Almanac and Book of Facts, 1975, page 406."
  ) %>%
  tab_source_note(
    source_note = md("Reference: McNeil, D. R. (1977) *Interactive Data Analysis*.
Wiley.")
  )
gt_tbl
```

This code will generate the following gt table:

Large Landmasses of the World

The top ten largest are presented

name	size
Asia	16988
Africa	11506
North America	9390
South America	6795
Antarctica	5500
Europe	3745
Australia	2968
Greenland	840
New Guinea	306
Borneo	280

Source: The World Almanac and Book of Facts, 1975, page 406.

Reference: McNeil, D. R. (1977) *Interactive Data Analysis*. Wiley.

You can learn more about the basics of gt using the following resources:

https://github.com/rich-iannone/presentations
https://themockup.blog/static/resources/gt-cookbook.html
https://themockup.blog/static/resources/gt-cookbook-advanced.html
https://gt.rstudio.com/articles/intro-creating-gt-tables.html

Simple Adverse Events Example

Below we will look at a simple gt example using adverse events data. The FDA maintains an API that houses a number of high-value, high priority and scalable structured datasets, including adverse events, drug product labeling, and recall enforcement reports.

https://open.fda.gov/

The R package openfda provides convenient access to the OpenFDA API. Data in the API are often changing as new data from hospitals, doctors, and other organizations are incorporated into the existing datasets.

A powerful part of gt is that it works well with the Tidyverse for data preparation. We can use Tidyverse packages such as dplyr and tidyr to filter, summarize, mutate, and reshape data tables and otherwise prepare them for gt. Please read for more information:

https://posit.co/blog/how-to-learn-r-as-a-sas-user/

The example below is available here:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/GT_Workflow.Rmd

This first part is creating a tidy tibble to pass into gt using openFDA data. The openfda functions are not displayed below and can be seen in the full example at the link above.

```
drug<- "Keytruda"
female<-get_adverse(2, drug) %>% mutate(sex = 'female')
male<-get_adverse(1, drug)%>% mutate(sex = 'male')

all<- total_events(drug)
total_count<- sum(all$count)
serious<-serious_events(drug) %>% filter(term==1) %>% pull(count)
```

Then we will bind the male and female tibbles into one table, modify some cell values, reshape the table into a wide form, add a column, and take 10 rows from that table. Given that it's ready for gt, we pass that into the gt() function and use functions from gt to build a presentation-ready table:

```
rbind(male,female) %>%
  mutate(term=str_to_title(term)) %>%
  pivot_wider(names_from=sex, values_from=count) %>%
  mutate(Total=female+male) %>%
  top_n(10, Total) %>%
  gt(rowname_col="term") %>% tab_header(
    title = md("Adverse Events"),
    subtitle = "The top ten most reported events"
) %>% tab_source_note("All data queried from openFDA") %>%
  tab_stubhead(label= 'Patient Reaction') %>%
  tab_spanner(label = "By sex",
              columns = c(female, male)) %>%
  tab_spanner(label= 'Total Cases',
```

```
            columns = Total) %>%
  fmt_number(columns = c(female, male, Total), sep_mark = ',', decimals=0) %>%
cols_label(Total = '',
           female = 'Female',
           male= 'Male')
```

With the collection of gt functions used above, we've created the following table:

Adverse Events

The top ten most reported events

Patient Reaction	By sex		Total Cases
	Female	Male	
Malignant Neoplasm Progression	2,608	3,608	6,216
Death	1,322	1,921	3,243
Off Label Use	963	1,262	2,225
Diarrhoea	1,294	1,171	2,465
Fatigue	1,487	1,121	2,608
Product Use In Unapproved Indication	1,141	1,062	2,203
Pyrexia	920	911	1,831
Rash	746	773	1,519
Decreased Appetite	726	745	1,471
Nausea	1,067	615	1,682

All data queried from openFDA

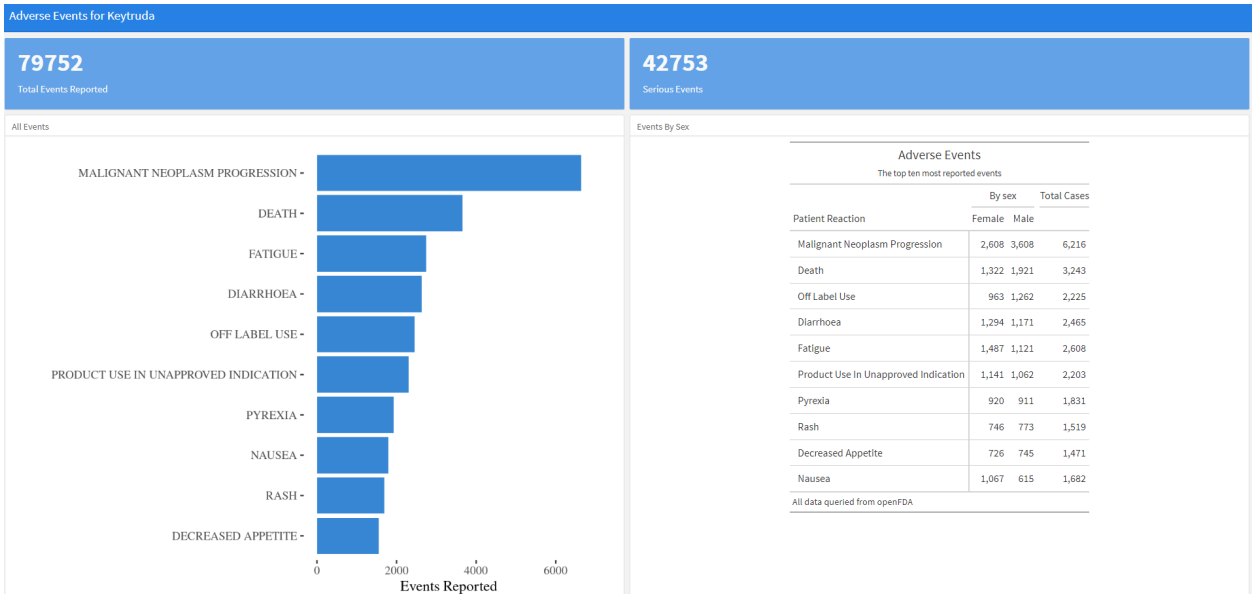
Please note that often gt tables will be presented in a literate programming tool such as Quarto.

https://quarto.org/

An example can be seen at the links below:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/philsdrugreport.Rmd

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/1-dashboard.Rmd



CDISC TLG - 3 EXAMPLES

In 2007/2008, Cathy Barrows, was the Co-Leader of CDISC Pilot Project to demonstrate that data submitted to the FDA using the CDISC Standard will meet the needs and expectations of both medical and statistical FDA reviewers. Gregory Steffens and Ian Fleming followed it with a Second CDISC Pilot Project focused on metadata.

The CDISC pilot package is real clinical trial data, provided by Eli Lilly and Company, de-identified & documents redacted. CDISC pilot package description:

Case Study Title: Safety and Efficacy of the Xanomeline Transdermal Therapeutic System (TTS) in Patients with Mild to Moderate Alzheimer's Disease
Indication: Alzheimer's Randomized, double-blind, placebo-controlled, parallel-group study
Three treatment arms: low dose, high dose, placebo
Approximately 300 patients, multiple centers

We will use the CDISC pilot package to replicate 3 submission outputs to highlight the functionality of gt for TLGs.

The tables include:

Table 14.2.01 "Summary of Demographic and Baseline Characteristics"
Table 14.3.13 "CIBIC+ - Categorical Analysis - LOCF"
Table 14-5.01 Incidence of Treatment Emergent Adverse Events by Treatment Group

For a list of examples beyond the 3 above, please see:

https://gt.rstudio.com/articles/case-study-clinical-tables.html

To learn more about the basics of creating clinical tables in R, please refer to the Tables in Clinical Trials with R book:

https://rconsortium.github.io/rtrs-wg/

The examples below work with ADaM data. In R, ADaM data are commonly prepared with the Pharmaverse package called Admiral. More information can be found at:

https://posit.co/blog/creating-adsl-with-the-pharmaverse-part-1/
https://posit.co/blog/pharmaverse-packages-for-clinical-reporting-workflows/

For SDTM, there is much excitement for OAK. OAK Garden is part of the "next-generation" solution for Roche's data and analytics platforms to move towards increased automation. The OAK Garden will be used by the Data Science teams to produce SDTM. More information is below:

https://www.cdisc.org/sites/default/files/2023-02/SDTM_Automation_OAK_CDISC_COSA.pdf
https://www.cdisc.org/oak
https://wiki.cdisc.org/pages/viewpage.action?pageId=179322489

Organizations also use the metacore package for metadata for the purpose of better enabling programming activities and functionality of other packages within the clinical programming workflow. metacore also works with define.xml.

https://github.com/atorus-research/metacore

For the tables below, the Data Prep mostly stays out of gt and is implemented via the packages referenced above. Each table will include a small amount of data prep that are focused on the requirements needed by gt such as pulling each of the *N* values for the required columns.

For each example below, we will export each to the RTF format at the end of the code.

Example 1 - Table 14.2.01 "Summary of Demographic and Baseline Characteristics"

For this first example, we will break apart each component that will generate the multi-page table below. Often gt examples are combined. Below is broken out for learning purposes.

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/gt-01-14_2_01.Rmd

Protocol: CDISCPILOT01 *Page 1 of 3*
Population: Intent-to-Treat

Table 14.2.01
Summary of Demographic and Baseline Characteristics

category	label	Placebo (N=86)	Xanomeline Low Dose (N=84)	Xanomeline High Dose (N=84)	Total (N=254)	p-value[1]
Age (y)	n	86	84	84	254	0.5934
	Mean	75.2	75.7	74.4	75.1	
	SD	8.59	8.29	7.89	8.25	
	Median	76.0	77.5	76.0	77.0	
	Min	52.0	51.0	56.0	51.0	
	Max	89.0	88.0	88.0	89.0	
	<65 yrs	14 (16%)	8 (10%)	11 (13%)	33 (13%)	0.1439
	65-80 yrs	42 (49%)	47 (56%)	55 (65%)	144 (57%)	
	>80 yrs	30 (35%)	29 (35%)	18 (21%)	77 (30%)	
Sex	n	86	84	84	254	0.1409
	Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)	
	Female	53 (62%)	50 (60%)	40 (48%)	143 (56%)	
Race (Origin)	n	86	84	84	254	0.6477
	Caucasian	75 (87%)	72 (86%)	71 (85%)	218 (86%)	
	African Descent	8 (9%)	6 (7%)	9 (11%)	23 (9%)	
	Hispanic	3 (3%)	6 (7%)	3 (4%)	12 (5%)	
	Other	0	0	1 (1%)	1 (1%)	
MMSE	n	86	84	84	254	0.5947
	Mean	18.0	17.9	18.5	18.1	
	SD	4.27	4.22	4.16	4.21	
	Median	19.5	18.0	20.0	19.0	
	Min	10.0	10.0	10.0	10.0	

[1] P-values are results of ANOVA treatment group comparison for continuous variable and Pearson's chisquare test for categorical variables.
Program Source: 14-2.01.R Executed: (Draft) 2022-05-20

Steps:

1. Add in a footnote from an external Excel file (here, called `titles.xlsx`).

```
```{r}
# Get the table that contains all text for different table types
titles <- readxl::read_xlsx("data/titles.xlsx")

# Extract the footnote text for the p-value column
p_footnote <-
  titles %>%
  filter(table_number == "14-2.01" & type == "footnote" & index == 1) %>%
  pull(text1)
```
```

2. Introduce the data frame to the `gt()` function and look at the table in the RStudio Viewer. It will be an HTML table but it will closely follow the structure and formatting of the final RTF table.

```
```{r}
gt_table <- gt(tbl)

gt_table
```
```

3. We will continue to build the gt table with functions. Next, we will add a table header. This contains a title (`"Table 14.2.01"`) and a subtitle (`"Summary of Demographic and Baseline Characteristics"`). We will also

provide two `preheader` lines that describe the protocol and the population. Note that these preheader lines won't appear in the HTML table preview but *will* appear in the final RTF document.

```{r}
gt_table <-
  gt_table %>%
  tab_header(
    title = "Table 14.2.01",
    subtitle = "Summary of Demographic and Baseline Characteristics",
    preheader = c("Protocol: CDISCPILOT01", "Population: Intent-to-Treat")
  )

gt_table
```

4. Let's remove all of those cells that have `NA` in them. We can do this with the `fmt_missing()` function. Ensure that all columns are targeted with `columns = everything()` and that the replacement text is an empty string (`""`).

```{r}
gt_table <-
  gt_table %>%
  fmt_missing(columns = everything(), missing_text = "")

gt_table
```

5. Let's do some formatting of cell values now. The `placebo`, `xanomeline_ld`, `xanomeline_hd`, and `total` columns have summary stats values that require decimal places for precision. All other values should be untouched (i.e., remain as integers). To do this, we're going to use `fmt_number()` on those columns but also use the `rows` argument to declare that only those cells of certain rows should get this type of formatting. The `label` column can be used to help with this, only those labels that refer to stats measures will have cells that need the formatting. In this case, we want three significant digits so we are going to use the `n_sigfig` argument with the value of `3`.

```{r}
gt_table <-
  gt_table %>%
  fmt_number(
    columns = c(placebo, xanomeline_ld, xanomeline_hd, total),
    rows = label %in% c("Mean", "SD", "Median", "Min", "Max"),
    n_sigfig = 3
  )

gt_table
```

6. The `p` column to the far right requires 4 decimal places for every single value. We can use `fmt_number()` for this, supplying the column name to `columns` and using `decimals = 4`.

```{r}
gt_table <-
  gt_table %>%
  fmt_number(columns = p, decimals = 4)

gt_table
```

7. Every column that ends in `_pct` is a percentage value but it needs a `%` sign. We can use `fmt_percent()` to add the percent sign easily to values in these columns. We don't have to name each of these columns, we

can use the `ends_with()` helper function and supply the text fragment `"_pct"` to target all of the relevant columns. In this case, we don't want decimals for any of the values (use `decimals = 0`), and, very importantly, we have to tell this function that these values are already in the correct form and don't need to be scaled (with `scale_values = FALSE`).

```{r}
gt_table <-
  gt_table %>%
  fmt_percent(
    columns = ends_with("_pct"),
    decimals = 0,
    scale_values = FALSE
  )

gt_table
```

8. There are groups of columns that contain N values and percentage values. But the convention is to use `N (x%)` in each cell. There is a function in gt that allows for columns to be merged and combined into a set format. For this case we need the `cols_merge_n_pct()` function to take these pairs of columns and merge them with the appropriate formatting.

```{r}
gt_table <-
  gt_table %>%
  cols_merge_n_pct(col_n = placebo, col_pct = placebo_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_ld, col_pct = xanomeline_ld_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_hd, col_pct = xanomeline_hd_pct) %>%
  cols_merge_n_pct(col_n = total, col_pct = total_pct)

gt_table
```

9. The column labels need labels that work better than the short ones commonly used for data analysis. With `cols_label()` we can replace the default labels (i.e., column names are used as the default labels). We wrap the label strings with `md()` to enable Markdown formatting. To add a line break, use the following bit of text `" \n"`. Note that we are pasting in data from variables we declared earlier.

```{r}
gt_table <-
  gt_table %>%
  cols_label(
    placebo  = md(paste0("Placebo  \n(N=", placebo_n, ")")),
    xanomeline_ld = md(paste0("Xanomeline  \nLow Dose  \n(N=", xanomeline_ld_n, ")")),
    xanomeline_hd = md(paste0("Xanomeline  \nHigh Dose  \n(N=", xanomeline_hd_n,
")")),
    total = md(paste0("Total  \n(N=", total_n, ")")),
    p = "p-value[1]"
  )

gt_table
```

10. Let's add in a footnote and a source note. This can be done with the `tab_footnote()` and `tab_source_note()` functions. In the final table, footnotes always appear before the source notes.

```{r}
gt_table <-
  gt_table %>%
  tab_footnote(footnote = p_footnote) %>%
```

```
  tab_source_note(
    source_note = paste('Program Source: 14-2.01.R     Executed: (Draft)',
reporting_date)
  )

gt_table
```

11. We need a few finishing touches to ensure the table presents well in RTF. If the default widths are not ideal, we can adjust them with the `cols_width()` function. You can use column indices or column names on the left-hand side (of the `~`). We can use `pct()` (percentage) or `px()` (pixel) values on the right-hand side. This may require some trial-and-error to get right (i.e., look at the RTF, adjust widths, look again, etc.).

    For Pharma RTF applications, we probably want to set some very specific page-layout options. These are found in the `tab_options()` function and four options are used here: `page.orientation`, `page.numbering`, `page.header.use_tbl_headings`, and `page.footer.use_tbl_notes`. We want `"landscape"` for the first and `TRUE` for all the rest.

```{r}
gt_table <-
  gt_table %>%
  cols_width(
    1 ~ pct(25),
    2 ~ pct(20)
  ) %>%
  tab_options(
    page.orientation = "landscape",
    page.numbering = TRUE,
    page.header.use_tbl_headings = TRUE,
    page.footer.use_tbl_notes = TRUE
  )
```

Putting it all together, you can write the table as one expression that uses `%>%` to link together the various directives.

```{r}
gt_table_final <-
  tbl %>%
  gt() %>%
  tab_header(
    title = "Table 14.2.01",
    subtitle = "Summary of Demographic and Baseline Characteristics",
    preheader = c("Protocol: CDISCPILOT01", "Population: Intent-to-Treat")
  ) %>%
  fmt_missing(columns = everything(), missing_text = "") %>%
  fmt_number(
    columns = c(placebo, xanomeline_ld, xanomeline_hd, total),
    rows = label %in% c("Mean", "SD", "Median", "Min", "Max"),
    n_sigfig = 3
  ) %>%
  fmt_number(columns = p, decimals = 4) %>%
  fmt_percent(
    columns = ends_with("_pct"),
    decimals = 0,
    scale_values = FALSE
  ) %>%
  cols_merge_n_pct(col_n = placebo, col_pct = placebo_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_ld, col_pct = xanomeline_ld_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_hd, col_pct = xanomeline_hd_pct) %>%
  cols_merge_n_pct(col_n = total, col_pct = total_pct) %>%
```

```
  cols_label(
    placebo  = md(paste0("Placebo  \n(N=", placebo_n, ")")),
    xanomeline_ld = md(paste0("Xanomeline  \nLow Dose  \n(N=", xanomeline_ld_n, ")")),
    xanomeline_hd = md(paste0("Xanomeline  \nHigh Dose  \n(N=", xanomeline_hd_n,
")")),
    total = md(paste0("Total  \n(N=", total_n, ")")),
    p = "p-value[1]"
  ) %>%
  tab_footnote(footnote = p_footnote) %>%
  tab_source_note(
    source_note = paste('Program Source: 14-2.01.R     Executed: (Draft)',
reporting_date)
  ) %>%
  cols_width(
    1 ~ pct(25),
    2 ~ pct(20)
  ) %>%
  tab_options(
    page.orientation = "landscape",
    page.numbering = TRUE,
    page.header.use_tbl_headings = TRUE,
    page.footer.use_tbl_notes = TRUE
  )
```

Writing the table to HTML can be done with `gtsave()`.

```
```{r}
gt_table_final %>% gtsave("html_14.2.01.html")
```
```

Write the **gt** table to an RTF document.

```
```{r}
gt_table_final %>% gtsave("tbl_14.2.01.rtf")
```
```

Example 2 - Table 14.3.13 "CIBIC+ - Categorical Analysis - LOCF"

The code for the second example is here:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/gt-02-14_3_13.Rmd

The second table will look like this in RTF:

Table 14.3.13
CIBIC+ - Categorical Analysis - LOCF

	Assessment	Placebo (N=79)	Xanomeline Low Dose (N=81)	Xanomeline High Dose (N=74)	p-value[1]
Week 8	n	77	81	73	0.2727
	Marked improvement	0	0	0	
	Moderate improvement	1 (1%)	2 (2%)	1 (1%)	
	Minimal improvement	19 (25%)	16 (20%)	13 (18%)	
	No Change	45 (58%)	48 (59%)	38 (52%)	
	Minimal worsening	10 (13%)	14 (17%)	20 (27%)	
	Moderate worsening	2 (3%)	1 (1%)	1 (1%)	
	Marked worsening	0	0	0	
Week 16	n	79	81	74	0.4003
	Marked improvement	0	0	0	
	Moderate improvement	0	3 (4%)	2 (3%)	
	Minimal improvement	12 (15%)	12 (15%)	13 (18%)	
	No Change	41 (52%)	46 (57%)	39 (53%)	
	Minimal worsening	25 (32%)	19 (23%)	20 (27%)	
	Moderate worsening	1 (1%)	1 (1%)	0	
	Marked worsening	0	0	0	
Week 24	n	79	81	74	0.6183
	Marked improvement	0	0	0	
	Moderate improvement	1 (1%)	1 (1%)	0	
	Minimal improvement	9 (11%)	14 (17%)	11 (15%)	

The code is below:

```{r}
gt_table_final <-
  tbl %>%
  gt() %>%
  tab_header(
    title = "Table 14.3.13",
    subtitle = "CIBIC+ - Categorical Analysis - LOCF",
    preheader = c("Protocol: CDISCPILOT01", "Population: Efficacy")
  ) %>%
  fmt_missing(columns = everything(), missing_text = "") %>%
  fmt_number(columns = p, decimals = 4) %>%
  fmt_percent(
    columns = ends_with("_pct"),
    decimals = 0,
    scale_values = FALSE
  ) %>%
  cols_merge_n_pct(col_n = placebo, col_pct = placebo_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_ld, col_pct = xanomeline_ld_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_hd, col_pct = xanomeline_hd_pct) %>%
  cols_label(
    category = "",
    label = "Assessment",
    placebo  = md(paste0("Placebo  \n(N=", placebo_n, ")")),
    xanomeline_ld = md(paste0("Xanomeline  \nLow Dose  \n(N=", xanomeline_ld_n, ")")),
    xanomeline_hd = md(paste0("Xanomeline  \nHigh Dose  \n(N=", xanomeline_hd_n,
")")),
    p = "p-value"
  ) %>%
  tab_footnote(
```

```
    footnote = "Overall comparison of treatments using CMH test (Pearson Chi-Square),
controlling for site group.",
    locations = cells_column_labels(columns = p)
  ) %>%
  tab_source_note(
    source_note = reporting_date
  ) %>%
  cols_width(
    1 ~ pct(10),
    2 ~ pct(25)
  ) %>%
  tab_options(
    page.orientation = "landscape",
    page.numbering = TRUE,
    page.header.use_tbl_headings = TRUE,
    page.footer.use_tbl_notes = TRUE
  )
```

Writing the table to HTML can be done with `gtsave()`.

```{r}
gt_table_final %>% gtsave("html_14.3.13.html")
```

Write the **gt** table to an RTF document.

```{r}
gt_table_final %>% gtsave("tbl_14.3.13.rtf")
```

Example 3 - Table 14-5.01 Incidence of Treatment Emergent Adverse Events by Treatment Group

The code for the third example is here:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/gt-03-14_5_01.Rmd

The third table adds some complexity. This table is built up using many of the same functions we used in the 2 examples above. There are some notable differences though:

- the `fmt_integer()` function is being used to format numeric values as integers, and, we are wrapping each value in square brackets with `pattern = "[{x}]"`

- there are two `fmt_numeric()` statements that format p-values and they use a `rows` expression (`fisher_placebo_v_xanomeline_ld < 0.2`) to target values less than `0.2` and the `pattern = "{x}*"` will apply the asterisk to the p-value in these cases

- there is another set of `fmt_numeric()` statements that replace values close to `1` with `>0.99` by using a method similar to the above (`rows` expression, and a `pattern`)

- the `tab_spanner()` function is used to create spanner column labels (these are labels over top column labels)

The third table will look like this in RTF:

Protocol: CDISCPILOT01 *Page 1 of 15*
Population: Safety

Table 14-5.01
Incidence of Treatment Emergent Adverse Events by Treatment Group

label	Placebo (N=86) n(%)	[AEs]	Xanomeline Low Dose (N=84) n(%)	[AEs]	Xanomeline High Dose (N=84) n(%)	[AEs]	Fisher's Exact p-values: Placebo vs. Low Dose	Fisher's Exact p-values: Placebo vs. High Dose
ANY BODY SYSTEM	65 (75.6%)	[281]	77 (91.7%)	[412]	76 (90.5%)	[433]	0.007*	0.014*
NA								
CARDIAC DISORDERS	12 (14.0%)	[26]	13 (15.5%)	[30]	15 (17.9%)	[30]	0.831	0.534
SINUS BRADYCARDIA	2 (2.3%)	[2]	7 (8.3%)	[10]	8 (9.5%)	[12]	0.097*	0.056*
MYOCARDIAL INFARCTION	4 (4.7%)	[4]	2 (2.4%)	[4]	4 (4.8%)	[8]	0.682	>.99
ATRIAL FIBRILLATION	1 (1.2%)	[1]	1 (1.2%)	[1]	3 (3.6%)	[5]	>.99	0.365
ATRIAL FLUTTER	0		1 (1.2%)	[1]	1 (1.2%)	[2]	0.494	0.494
CARDIAC DISORDER	0		0		1 (1.2%)	[1]		0.494
SUPRAVENTRICULAR EXTRASYSTOLES	1 (1.2%)	[2]	1 (1.2%)	[2]	1 (1.2%)	[1]	>.99	>.99
VENTRICULAR EXTRASYSTOLES	0		2 (2.4%)	[4]	1 (1.2%)	[1]	0.243	0.494
ATRIAL HYPERTROPHY	1 (1.2%)	[2]	0		0		>.99	>.99
ATRIOVENTRICULAR BLOCK FIRST DEGREE	1 (1.2%)	[1]	1 (1.2%)	[1]	0		>.99	>.99
ATRIOVENTRICULAR BLOCK SECOND DEGREE	1 (1.2%)	[1]	0		0		>.99	>.99
BRADYCARDIA	1 (1.2%)	[4]	0		0		>.99	>.99
BUNDLE BRANCH BLOCK LEFT	1 (1.2%)	[1]	0		0		>.99	>.99
BUNDLE BRANCH BLOCK RIGHT	1 (1.2%)	[2]	1 (1.2%)	[1]	0		>.99	>.99
CARDIAC FAILURE CONGESTIVE	1 (1.2%)	[1]	0		0		>.99	>.99
PALPITATIONS	0		2 (2.4%)	[2]	0		0.243	
SINUS ARRHYTHMIA	1 (1.2%)	[2]	0		0		>.99	>.99

Note: Treatment emergent events are defined as events which start on or after the start of treatment.
Note: Adverse events are coded using MedDRA.
Note: Percentages are based on the number of subjects in the safety population within each treatment group.
Note: P-values are based on Fisher's Exact test for the comparison of placebo versus each active treatment group. An asterisk is appended to p-values that are less than 0.15.
Note: The column [AE] represents the total number of times an event was recorded.
Program Source: t-14-5-01.R Executed: (Draft) 2022-05-16

The code is below:

```r
tbl <-
  tbl %>%
  mutate(label = paste0("  ", label))

gt_table_final <-
  tbl %>%
  gt() %>%
  tab_header(
    title = "Table 14-5.01",
    subtitle = "Incidence of Treatment Emergent Adverse Events by Treatment Group",
    preheader = c("Protocol: CDISCPILOT01", "Population: Safety")
  ) %>%
  fmt_missing(columns = everything(), missing_text = "") %>%
  fmt_number(
    columns = c(fisher_placebo_v_xanomeline_ld, fisher_placebo_v_xanomeline_hd),
    decimals = 4
  ) %>%
  fmt_percent(
    columns = ends_with("_pct"),
    decimals = 1,
    scale_values = FALSE
  ) %>%
  fmt_integer(
```

```
    columns = ends_with("_ae"),
    pattern = "[{x}]"
  ) %>%
  fmt_number(
    columns = starts_with("fisher"),
    decimals = 3
  ) %>%
  fmt_number(
    columns = fisher_placebo_v_xanomeline_ld,
    rows = fisher_placebo_v_xanomeline_ld < 0.2,
    decimals = 3,
    pattern = "{x}*"
  ) %>%
  fmt_number(
    columns = fisher_placebo_v_xanomeline_hd,
    rows = fisher_placebo_v_xanomeline_hd < 0.2,
    decimals = 3,
    pattern = "{x}*"
  ) %>%
  fmt_number(
    columns = fisher_placebo_v_xanomeline_ld,
    rows = fisher_placebo_v_xanomeline_ld > 0.99,
    pattern = ">.99"
  ) %>%
  fmt_number(
    columns = fisher_placebo_v_xanomeline_hd,
    rows = fisher_placebo_v_xanomeline_hd > 0.99,
    pattern = ">.99"
  ) %>%
  cols_merge_n_pct(col_n = placebo_n, col_pct = placebo_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_ld_n, col_pct = xanomeline_ld_pct) %>%
  cols_merge_n_pct(col_n = xanomeline_hd_n, col_pct = xanomeline_hd_pct) %>%
  tab_spanner(
    label = md(paste0("Placebo  \n(N=", placebo_n, ")")),
    columns = starts_with("placebo")
  ) %>%
  tab_spanner(
    label = md(paste0("Xanomeline  \nLow Dose  \n(N=", xanomeline_ld_n, ")")),
    columns = starts_with("xanomeline_ld")
  ) %>%
  tab_spanner(
    label = md(paste0("Xanomeline  \nHigh Dose  \n(N=", xanomeline_hd_n, ")")),
    columns = starts_with("xanomeline_hd")
  ) %>%
  tab_spanner(
    label = md("Fisher's Exact  \np-values"),
    columns = starts_with("fisher")
  ) %>%
  cols_label(
    placebo_n = "n(%)",
    xanomeline_ld_n = "n(%)",
    xanomeline_hd_n = "n(%)",
    placebo_ae = "[AEs]",
    xanomeline_ld_ae = "[AEs]",
    xanomeline_hd_ae = "[AEs]",
    fisher_placebo_v_xanomeline_ld = md("Placebo  \nvs.  \nLow Dose"),
    fisher_placebo_v_xanomeline_hd = md("Placebo  \nvs.  \nHigh Dose")
  ) %>%
  tab_footnote(footnote = "Note: Treatment emergent events are defined as events which
start on or after the start of treatment.") %>%
  tab_footnote(footnote = "Note: Adverse events are coded using MedDRA.") %>%
  tab_footnote(footnote = "Note: Percentages are based on the number of subjects in
the safety population within each treatment group.") %>%
```

```
  tab_footnote(footnote = "Note: P-values are based on Fisher's Exact test for the
comparison of placebo versus each active treatment group. An asterisk is appended to
p-values that are less than 0.15.") %>%
  tab_footnote(footnote = "Note: The column [AE] represents the total number of times
an event was recorded.") %>%
  tab_source_note(
    source_note = paste('Program Source: t-14-5-01.R     Executed: (Draft)',
reporting_date)
  ) %>%
  cols_width(
    1 ~ pct(30),
    ends_with("n") ~ pct(34/3),
    ends_with("ae") ~ pct(18/3),
    starts_with("fisher") ~ pct(18/2)
  ) %>%
  tab_options(
    page.orientation = "landscape",
    page.numbering = TRUE,
    page.header.use_tbl_headings = TRUE,
    page.footer.use_tbl_notes = TRUE
  )
```

Writing the table to HTML can be done with `gtsave()`.

```{r}
gt_table_final %>% gtsave("html_14.5.01.html")
```

Write the **gt** table to an RTF document.

```{r}
gt_table_final %>% gtsave("tbl_14.5.01.rtf")
```

Future Considerations - interactive TLGs, ARDs & Shiny

Many pharmaceutical companies are interested in using interactivity and or Shiny for clinical trials, either for internal analysis, generating TLGs or for inclusion within a submission to the regulatory agency. Roche built a framework called teal that leverages the R Shiny package to scale development of our shiny apps. You can learn more about teal below:

https://posit.co/blog/roche-shifting-to-an-open-source-backbone-in-clinical-trials/

https://pharmaverse.org/e2eclinical/tlg/

Tables will be an important component of these applications. The Shiny for Submissions Task Force meets every quarter to discuss topics around using Shiny for submissions:

https://learn.rinpharma.com/shiny4submissions.html

Below are examples of each use-case:

1. **Internal Analysis Example**

Biogen created tidyCDISC as a shiny app to easily create custom tables and figures from ADaM-ish data sets. Its source code is here:

https://github.com/Biogen-Inc/tidyCDISC

You can see it live here:

https://rinpharma.shinyapps.io/tidyCDISC/

The FDA also uses Shiny internally as seen here:

https://www.lexjansen.com/phuse-us/2020/dv/DV07.pdf

2. **Shiny generating TLGs**

William Noble (Sarah Cannon Research Institute) created a shiny app called the “Future of clinical tfls” here:

https://williamnoble.shinyapps.io/the_future_of_clinical_tfls/

3. **Shiny app in a Submission Package**

The R FDA Pilot included a Shiny app that was accepted by the FDA.

https://github.com/RConsortium/submissions-pilot2

You can read more about it here:

https://colorado.posit.co/rsc/Shiny-for-Submissions/template.html#/title-slide

Below we will review the steps to add gt to Shiny. Below is a Shiny example using gt.

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/tree/master/shiny_gt

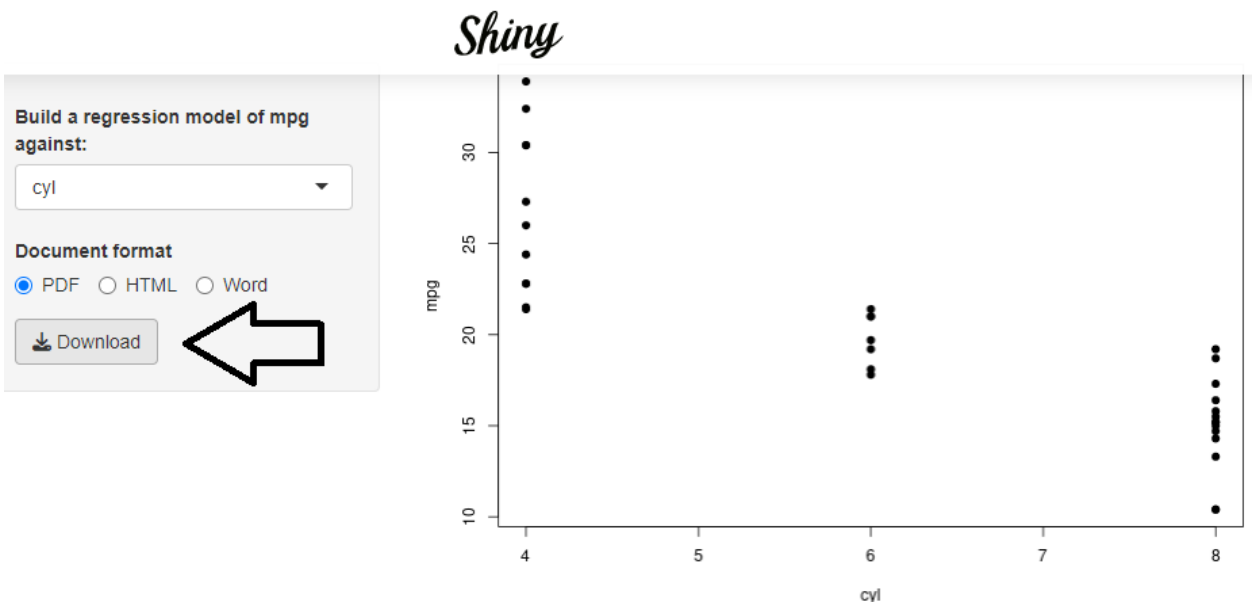
In the example above, the Shiny app will call a R Markdown/Quarto file via the Server function:

```
content = function(file) {
      src <- normalizePath('report.Rmd')

      # temporarily switch to the temp dir, in case you do not have write
      # permission to the current working directory
      owd <- setwd(tempdir())
      on.exit(setwd(owd))
      file.copy(src, 'report.Rmd', overwrite = TRUE)

      library(rmarkdown)
      out <- render('report.Rmd', switch(
        input$format,
        PDF = pdf_document(), HTML = html_document(), Word = word_document()
      ))
      file.rename(out, file)
    }
```

The report.Rmd contains the gt code. When the Download button is clicked, it will generate a pdf (or RTF) report containing a gt table.

The report code is here:

https://github.com/philbowsher/Clinical-Tables-in-R-with-gt/blob/master/shiny_gt/report.Rmd



You can see a live similar example here:

https://shiny.rstudio.com/gallery/download-knitr-reports.html

You can learn more about Shiny here:

https://mastering-shiny.org/

It will be also interesting to watch how analysis results standards impact/change the reporting methods by pharmas. More can be found here:

https://www.cdisc.org/standards/foundational/analysis-results-standards

Contact & Summary

The information above highlights the exciting gt R package and example use cases, and how well established tools like these can help modernize clinical processes for reporting via reports, dashboards and Shiny applications.

Phil Bowsher

phil@posit.co

https://github.com/philbowsher