Study Notes: `gt` package

Jihong Zhang

2020-05-25

Table of contents

Overview

This post is inspired by themockup's blog and RStudio's documentation of `gt` package[1].

> "We can construct a wide variety of useful tables with a cohesive set of table parts. These include the table header, the stub, the column labels and spanner column labels, the table body, and the table footer."

Component

[1] You can find the documentation here.

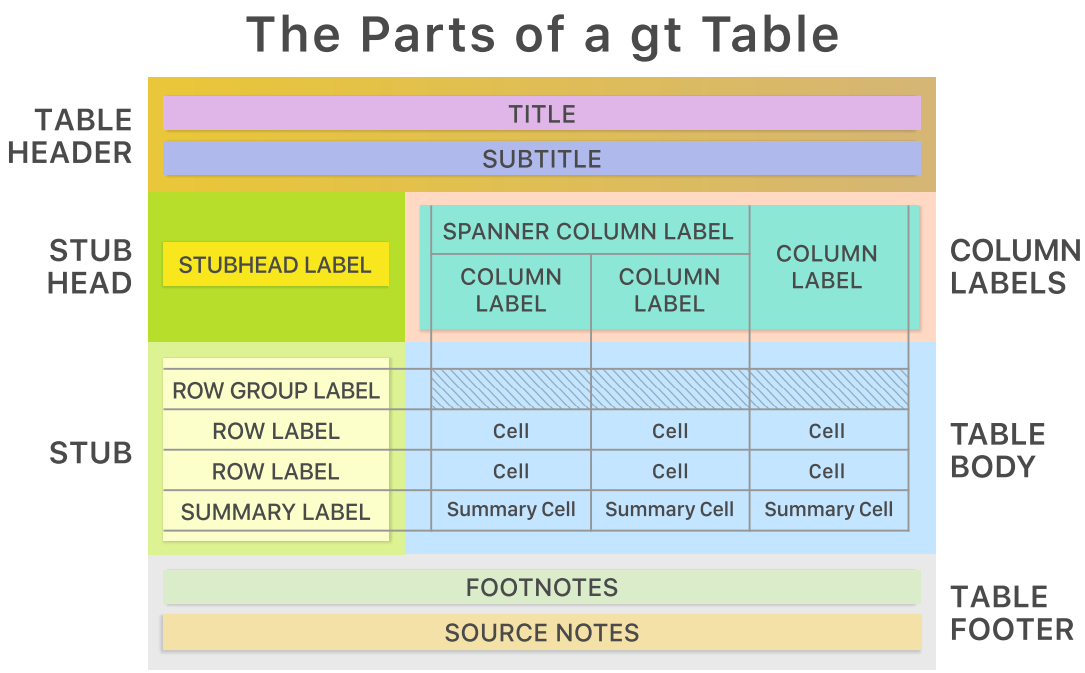

Figure 1: The Parts of a gt Table

The typical gt table starts with converting a data frame into a gt object. The `gt` object is then modified by adding various components to it. The components include the table header, the stub, the column labels and spanner column labels, the table body, and the table footer.

Example: sp500 data

As always, we install and load `gt` and `tidyverse` packages.

```
library(gt)
library(tidyverse)
```

- `gt()` function can convert data.frame into `gt` object. A `gt` object can be directly output and rendered into HTML.

```
# Define the start and end dates for the data range
start_date <- "2010-06-07"
end_date <- "2010-06-14"

# Create a gt table based on preprocessed
# `sp500` table data
sp500_gt <- sp500 |>
  dplyr::filter(date >= start_date & date <= end_date) |>
  dplyr::select(-adj_close) |>
  gt()
```

```
sp500_gt
```

date	open	high	low	close	volume
2010-06-14	95.01	101.59	89.01	92.56	0
2010-06-11	82.61	92.21	77.11	91.96	0
2010-06-10	85.71	87.81	85.71	86.48	0
2010-06-09	72.71	77.71	72.21	75.83	0
2010-06-08	50.81	63.11	42.11	62.07	0
2010-06-07	65.81	71.31	49.81	60.74	0

`tab_header`: title and subtitle

- The `gt` object can be further modified by adding components to it, for example `tab_header` adds Table Headers (including title and subtitle).
  - For narratives like *title* or *subtitle*, you can also use markdown format with the `md` function.

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  )
```

S&P 500

June 7-14, 2010

date	open	high	low	close	volume
2010-06-14	95.01	101.59	89.01	92.56	0
2010-06-11	82.61	92.21	77.11	91.96	0
2010-06-10	85.71	87.81	85.71	86.48	0
2010-06-09	72.71	77.71	72.21	75.83	0
2010-06-08	50.81	63.11	42.11	62.07	0
2010-06-07	65.81	71.31	49.81	60.74	0

> **ℹ Note**
>
> Note that if your Quarto HTML has CSS style, the markdown format in `gt` object may not work as expected. Like the following title will be **bold** but with underline (my CSS style).

```r
sp500_gt |>
  tab_header(
    title = md("**S&P 500**"),
    subtitle = md("*June 7-14*, 2010")
  )
```

S&P 500

June 7-14, 2010

date	open	high	low	close	volume
2010-06-14	1095.00	1105.91	1089.03	1089.63	4425830000
2010-06-11	1082.65	1092.25	1077.12	1091.60	4059280000
2010-06-10	1058.77	1087.85	1058.77	1086.84	4144180000
2010-06-09	1062.75	1077.74	1052.25	1055.69	5983200000
2010-06-08	1050.81	1063.15	1042.17	1062.00	6192750000
2010-06-07	1065.84	1071.36	1049.86	1050.47	5467560000

`fmt_<column_type>`: format columns

- `fmt_<column_type>` functions can be used to format the columns of the table.
  - For example, `fmt_currency` can be used to format the `close` column as currency.
  - `fmt_date` can be used to format the `date` column as date.
  - `fmt_number` can be used to format the `volume` column as a number.

```r
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  ) |>
  fmt_currency(
    columns = vars(c(open, high, low, close)),
    currency = "USD"
  ) |>
  fmt_date(
```

```
    columns = vars(date),
    date_style = "wday_month_day_year"
  ) |>
  fmt_number(
    columns = vars(volume),
    suffixing = TRUE
  )
```

S&P 500

June 7-14, 2010

date	open	high	low	close	volume
Monday, June 14, 2010	$1,095.91	$1,105.91	$1,089.91	$1,089.63	4.43B
Friday, June 11, 2010	$1,082.61	$1,092.21	$1,077.11	$1,091.60	4.06B
Thursday, June 10, 2010	$1,058.71	$1,087.71	$1,058.71	$1,086.84	5.14B
Wednesday, June 9, 2010	$1,062.71	$1,077.71	$1,052.21	$1,055.69	5.98B
Tuesday, June 8, 2010	$1,050.81	$1,063.11	$1,042.11	$1,062.00	6.19B
Monday, June 7, 2010	$1,065.81	$1,071.81	$1,049.81	$1,050.47	5.47B

tab_source_note: source note

- `tab_source_note` can be used to add a source note to the table underneath the table.

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  ) |>
  tab_source_note(
    source_note = "Data from Yahoo Finance"
  )
```

S&P 500

June 7-14, 2010

date	open	high	low	close	volume
2010-06-11	1094.01	1105.91	1089.01	1094.82	3000000
2010-06-10	1082.61	1092.21	1077.11	1091.91	2880000
2010-06-08	1087.71	1087.71	1074.41	1078.81	4480000
2010-06-07	1062.71	1075.71	1052.21	1075.31	4290000
2010-06-04	1050.81	1063.11	1042.11	1062.21	4705000
2010-06-03	1065.81	1065.81	1045.41	1045.41	4576000

Data from Yahoo Finance

`tab_footnote`: footnotes

Beside the markdown format, Quarto HTML also accepts HTML code using `htmltools::p` function. But it may only apply to HTML not PDF. As Figure 1 shows, footnotes are located at the bottom of the table but at the top of *source notes*.

> **`cells_*()`: target cells in the table**
>
> We were able to supply the reference locations in the table by using the cells_body() helper function and supplying the necessary targeting through the columns and rows arguments. Other cells_*() functions have similar interfaces and they allow us to target cells in different parts of the table.

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
```

```
    subtitle = "June 7-14, 2010"
  ) |>
  tab_source_note(
    source_note = htmltools::p(align="right", "Data from Yahoo Finance")
  ) |>
  tab_footnote(
    footnote = "All values are in USD.",
    locations = cells_body(
      columns = vars(open),
      rows = date == "2010-06-14"
    )
  )
```

`tab_row_group`: row groups

- We can make a new row group with each tab_row_group() call.
- The inputs are row group names in the label argument, and row references in the rows argument.
- key arguments:
  - `label`: the name of the row group.
  - `rows`: the logical vector that specifies which rows belong to the group.

> **Note**
>
> Note that the sequence of adding `tab_row_group` will affect the order of the row groups in the final table. *Lastest added group will be on the top.*

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  ) |>
  tab_row_group(
    label = "Last three days",
    rows = date %in% c("2010-06-10", "2010-06-11", "2010-06-14")
  ) |>
  tab_row_group(
    label = "First three days",
    rows = date %in% c("2010-06-07", "2010-06-08", "2010-06-09")
  )
```

S&P 500

June 7-14, 2010

date	open	high	low	close	volume
First three days					
2010-06-09	1062.75	1077.74	1052.25	1055.69	5983200000
2010-06-08	1050.81	1063.15	1042.17	1062.00	6192750000
2010-06-07	1065.84	1071.36	1049.86	1050.47	5467560000
Last three days					
2010-06-14	1095.00	1105.91	1089.03	1089.63	4425830000
2010-06-11	1082.65	1092.25	1077.12	1091.60	4059280000
2010-06-10	1058.77	1087.85	1058.77	1086.84	5144780000

`tab_spanner`: column groups and spann column labels

- key arguments:
  - `label`: the name of the column group.
  - `columns`: the columns that belong to the group.

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  ) |>
  tab_spanner(
    label = "Price",
    columns = vars(open, high, low, close)
  ) |>
  tab_spanner(
    label = "Volume",
    columns = vars(volume)
  )
```

S&P 500

June 7-14, 2010

	Price				Volume
date	open	high	low	close	volume
2010-06-11	1095.01	1105.91	1089.01	1094.84	3,000,000,000
2010-06-11	1082.61	1092.22	1077.12	1091.51	3,000,000,000
2010-06-10	1058.71	1078.15	1058.71	1076.48	3,000,000,000
2010-06-09	1062.71	1077.71	1052.21	1058.36	3,000,000,000
2010-06-08	1050.81	1063.11	1042.11	1062.20	3,000,000,000
2010-06-07	1065.81	1071.31	1049.81	1050.71	3,000,000,000

`summary_rows`: summary rows

- `summary_rows` can be used to add summary rows to the table by `group`.
  - if you want to have grant summary statistics, use `grand_summary_rows` instead.
- key arguments:
  - `fns`: a list of functions to apply to the columns.
  - `columns`: the columns to which the functions are applied.

> **Note**
>
> Note that the `fmt` argument is used to format the summary values with a list. There are multiple formatting functions for numeric variables available in the gt package:
>
> - `fmt_number(cols = vars(...), decimals = 2)`: format numbers with a specified number of 2 decimal places.
> - `fmt_scientific(cols = vars(...), decimals = 3)`: format numbers in scientific notation with a specified number of 3 decimal places.

```
sp500_gt |>
  tab_header(
    title = "S&P 500",
    subtitle = "June 7-14, 2010"
  ) |>
  grand_summary_rows(
    fns = list(
      "Mean" = ~ mean(., na.rm =TRUE),
      "SD" = ~ sd(., na.rm =TRUE)
```

```
    ),
    columns = vars(open, high, low, close, volume),
    fmt = list(
      ~ fmt_scientific(data = .,
                       columns = vars(volume),
                       decimals = 3),
      ~ fmt_number(data = .,
                   columns = vars(open, high, low, close),
                   decimals = 2)
    )
  )
```

S&P 500

June 7-14, 2010

	date	open	high	low	close	volume
2010-	06-1140 95.0	10105.91	1089.01	43048295	6833	0000
2010-	06-1110 82.6	15092.21	5077.11	42009519	6208	0000
2010-	06-1100 58.7	17087.81	5058.71	57018464	8748	0000
2010-	06-0190 62.7	15077.71	4052.21	55095853	6290	0000
2010-	06-0180 50.8	11063.11	5042.11	67016922	0705	0000
2010-	06-0170 65.8	14071.31	6049.81	56045607	4576	0000
Mean	1,069.3	1,083.0	1,061.5	1,305.2	7.10	× 10^9
SD	—	16.41	15.43	17.91	18.84.56	× 10^8

`cols_xxx()`: manupulate columns' positions

- Functions:
  - `cols_move_to_start()`: move columns to the start of the table.
  - `cols_move_to_end()`: move columns to the end of the table.
  - `cols_hide()`: hide columns.

Styling of the table

A basic `gt` table can be created as so

```
data("iris")
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.…
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.…
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.…
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.…
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s…
```

You can add row names (`rowname_col` argument) and add group names (`groupname_col` argument) into the table:

```
iris_gt <- iris |>
  arrange(desc(Sepal.Length)) |> # 6 types of iris with largest sepal length
  mutate(Rank = 1:nrow(iris)) |>
  slice_head(n = 3, by = Species) |> # select top 3 Sepal length of each species
  gt(groupname_col = "Species", rowname_col = "Rank")
iris_gt
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica				
1	7.9	3.8	6.4	2.0
2	7.7	3.8	6.7	2.2
3	7.7	2.6	6.9	2.3
versicolor				
13	7.0	3.2	4.7	1.4
14	6.9	3.1	4.9	1.5
18	6.8	2.8	4.8	1.4
setosa				
71	5.8	4.0	1.2	0.2
78	5.7	4.4	1.5	0.4
79	5.7	3.8	1.7	0.3

`tab_style`: change style of cells

- `tab_style(data, style, locations)` allows you to change the style (`style`) of cells (`locations`) in the table.

Style functions for `style` argument

- the background color of the cell (`cell_fill()`: color)
- the cell's text color, font, and size (`cell_text()`: color, font, size)
- the text style (`cell_text()`: style), enabling the use of italics or oblique text.
- the text weight (`cell_text()`: weight), allowing the use of thin to bold text (the degree of choice is greater with variable fonts)
- the alignment and indentation of text (`cell_text()`: align and indent)
- the cell borders (`cell_borders()`)

```
iris_gt_colored <- iris_gt |>
  tab_style( # style for virginica
    style = list(
      cell_fill(color = "lightblue"),
      cell_text(weight = "bold")
    ),
    locations = cells_body(
        columns = colnames(iris),
        rows = Species == "virginica")
  ) |>
  tab_style( # style for versicolor
    style = list(
      cell_fill(color = "royalblue"),
      cell_text(color = "red", weight = "bold")
    ),
    locations = cells_body(
        columns = colnames(iris),
        rows = Species == "versicolor")
  )
iris_gt_colored
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica				
1	7.9	3.8	6.4	2.0
2	7.7	3.8	6.7	2.2
3	7.7	2.6	6.9	2.3
versicolor				
1 3	7.0	3.2	4.7	1.4
1 4	6.9	3.1	4.9	1.5
1 8	6.8	2.8	4.8	1.4
setosa	5.8	4.0	1.2	0.2

- Next, the boarder could be added into the table:

```
iris_gt_colored |>
  tab_style( # tab_style to change style of cells,
    # cells_borders provides the formatting
    # locations tells it where add black borders to all column labels
    style = cell_borders(
        sides = "left",
        color = "black",
        weight = px(1.2)
    ),
    locations = cells_body(columns = colnames(iris))
  ) |>
  # Add botton line below the column names
  tab_style(
    style = cell_borders(
        sides = "bottom",
        color = "black",
        weight = px(3)
    ),
    locations = cells_column_labels(columns = gt::everything())
  )
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica				
1	7.9	3.8	6.4	2.0
2	7.7	3.8	6.7	2.2
3	7.7	2.6	6.9	2.3
versicolor				
13	7.0	3.2	4.7	1.4
14	6.9	3.1	4.9	1.5
18	6.8	2.8	4.8	1.4
setosa				
71	5.8	4.0	1.2	0.2
78	5.7	4.4	1.5	0.4
79	5.7	3.8	1.7	0.3

`tab_options`: table output options

- `tab_options` allows you to change the output options of the table. There are multiple options available:

1. `table.font.names=` allows you to change the font of the table.

```
iris_gt_colored |>
  tab_options(
    table.font.names = c("Tenor Sans")
  )
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica				
1	7.9	3.8	6.4	2.0
2	7.7	3.8	6.7	2.2
3	7.7	2.6	6.9	2.3
versicolor				
13	7.0	3.2	4.7	1.4
14	6.9	3.1	4.9	1.5
18	6.8	2.8	4.8	1.4
setosa				
71	5.8	4.0	1.2	0.2
78	5.7	4.4	1.5	0.4
79	5.7	3.8	1.7	0.3

2. `table.width=` allows you to change the width of the table.

```
iris_gt_colored |>
  tab_options(table.width = px(700))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica				
1	7.9	3.8	6.4	2.0
2	7.7	3.8	6.7	2.2
3	7.7	2.6	6.9	2.3
versicolor				

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
13	7.0	3.2	4.7	1.4
14	6.9	3.1	4.9	1.5
18	6.8	2.8	4.8	1.4
setosa				
71	5.8	4.0	1.2	0.2
78	5.7	4.4	1.5	0.4
79	5.7	3.8	1.7	0.3