
Optimizing gt Tables for Enhanced Visualization

GRAPH Network & WHO, supported by the Global Fund to fight HIV, TB & Malaria

October 2023

This document is a draft of a lesson made by the GRAPH Network, a non-profit headquartered at the University of Geneva Global Health Institute, in collaboration with the World Health Organization, under a Global Fund 2023 grant to create e-learning modules to build in-country data capacity for epidemiological and impact analysis for National HIV, TB and malaria programs

Introduction	
Learning objectives	
Packages	
Previously in pt1	
Dataset	
Themes	
Formatting the values in the table	
Conditional formatting	
Fonts and text	
Borders	
Answer Key	
External resources and packages	

Introduction

The previous `gt` lesson focused mainly on the components of the table its structure and how to manipulate it properly. This lesson, presenting the second part of the `gt` series will focus on using the package to polish, style, and customize the visual effects of tables in a way that elevate the quality and efficiency of your reports.

Let's dig in.

Learning objectives

- Cells Formatting
- Conditional coloring
- Format text(font color, bold,etc.)
- Add borders to text

By the conclusion of this lesson, you will have the skills to artfully style your `gt` tables to meet your specific preferences achieving a level of detail similar to this:

Sum of HIV cases in Malawi				
from Q1 2019 to Q2 2019				
period	New cases		Previous cases	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Data from the Malawi HIV Program

Packages

In this lesson, we will use the following packages:

- `gt`
- `dplyr`, `tidyr`, and `purrr`.
- `janitor`
- `KableExtra`
- `Paletteer`, `ggsci`

```
pacman::p_load(tidyverse, janitor, gt, here)
```

Previously in pt1

RECAP



In the previous `gt` lesson we had the opportunity to :

- Discover the HIV prevalence data of Malawi.
- Discover the grammar of tables and the `gt` package.
- create simple table.
- Add details like title and footnote to the table.
- Group columns into spanners.
- Create Summary rows.

RECAP



The Parts of a gt Table

TABLE HEADER	TITLE			
	SUBTITLE			
STUB HEAD	STUBHEAD LABEL	SPANNER COLUMN LABEL		COLUMN LABEL
		COLUMN LABEL	COLUMN LABEL	
STUB	ROW GROUP LABEL			
	ROW LABEL	Cell	Cell	Cell
	ROW LABEL	Cell	Cell	Cell
	SUMMARY LABEL	Summary Cell	Summary Cell	Summary Cell
FOOTNOTES				
SOURCE NOTES				

COLUMN LABELS

TABLE BODY

TABLE FOOTER

Dataset

In this lesson, we will use the same data from the previous lesson, you can go back for a detailed description of the data and the preparation process we made.

RECAP



Here's the full details of the columns we will use:

RECAP



- **region:** The geographical region or area where the data was collected or is being analyzed.
- **period:** A specific time period associated with the data, often used for temporal analysis.
- **previous_negative:** The count or number of individuals with a previous negative test result.
- **previous_positive:** The count or number of individuals with a previous positive test result.
- **new_negative:** The count or number of newly diagnosed cases with a negative result.
- **new_positive:** The count or number of newly diagnosed cases with a positive result.

But for the purposes of this lesson we will use the tables directly, this is the table that we created with the right spanners and columns labels, we will base the rest of our lesson on this particular one.

```
hiv_malawi_summary <- read_rds(here::here("data", "clean",  
  "malawi_hiv_summary_t3.rds"))
```

```
hiv_malawi_summary
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Themes

Since the objective of this lesson is mainly styling, let's start with using a pre-defined theme to add more visuals and colors to the table and its components. To do so we use the `opt_stylize` function. The function contains multiple pre-defined styles and can accept a color as well. In our case we chose to go with style No.6 and the color 'cyan', you can set these arguments to your liking.

```
t1 <- hiv_malawi_summary %>%  
  opt_stylize(  
    style = 6,  
    color = 'gray'  
  )  
t1
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

CHALLENGE



For more sophisticated themes and styling, you can refer to the function `tab_options` (documentations [here](#)) which is basically the equivalent to the `theme` function in `ggplot2`. This function contains arguments and options on every single layer and component of the table. For the purposes of this lesson we won't dive into it.

Formatting the values in the table

Wouldn't it be useful to visualize in colors the difference between values in a specific column? In many reports, these kind of tables are quite useful especially if the number of rows is quite large. Let's do this for our table such that we have the `new_positive` column is formatted red.

```
# revised the documentation, paletteer is not needed.  
# my_colors <- paletteer::paletteer_d("ggsci::red_material")
```

We can do this by means of the `data_color` function for which we need two specify tow arguments, `columns` (as in at what column this styling will take place?) and `palette` as the color palette we intend to use.

```
t2 <- t1 %>%  
  data_color(  
    columns = new_positive, # the column or columns as we will see later  
    palette = "ggsci::red_material" # the palette form the ggsci package.  
  )  
t2
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

SIDE NOTE



`ggsci::red_material` is not the only palette we can use, in fact there are hundreds of palettes that are designed to be used in R. You can find a lot more in the `paletteer` package documentations in [here](#), or in the official `data_color` documentation [here](#).

We can do this for the `previous_negative` column as well. We can use a different kind of palette, I'm using for this case the green palette from the same package: `ggsci::green_material`, the palette you choose is a matter of convenience and personal taste, you can explore more about this if you refer to the side note above.

```
t2 %>%
  data_color(
    columns = previous_negative,
    palette = "ggsci::green_material"
  )
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Similarly, we can also color multiple columns at once, for example we can style the columns with positive cases in red, and those with negative cases in green. To do this we need to write *two* `data_color` statements one for each color style:

```
t4 <- t1 %>%
  data_color(
    columns = ends_with("positive"), # selecting columns ending with the word
    positive
    palette = "ggsci::red_material" # red palette
  ) %>%
  data_color(
    columns = ends_with("negative"), # selecting columns ending with the word
    negative
    palette = "ggsci::green_material" # green palette
  )
t4
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

REMINDER



Remember in the previous lesson we used the `tidyselect` functions to select columns, in the code above we used the function `ends_with` to select the columns ending either with the word 'negative' or 'positive' which is perfect for the purpose of our table.

Again, the column labels in the `gt` table and the actual column names in the `data.frame` can be different, in our case we refer to the names in the data.

Conditional formatting

We can also set up the table to conditionally change the style of a cell given its value. In our case we want to highlight values in the column `previous_positive` according to a threshold (the value 15700). Greater or equal values than the threshold should be in green.

To achieve this we use the `tab_style` function where we specify two arguments:

- `style`: where we specify the color in the `cell_text` function since we intend to manipulate the text within the cells.
- `location`: where we specify the columns and the rows of our manipulation in the `cells_body` since these cells are in the main body of the table.

```
t5 <- t4 %>%
  tab_style(
    style = cell_text(
      color = "red",
    ),
    locations = cells_body(
      columns = previous_positive,
      rows = previous_positive >= 15700
    )
  )
t5
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

WATCH OUT



In the code above, the condition over which the styling will occur is stated in :

```
locations = cells_body(columns = previous_positive, rows =
previous_positive >= 15700 )
```

Also, note that we can pass more arguments to the `cell_text` function, such as the size and the font of the cells we intend to style.

What if we want to have a two sided condition over the same threshold? Can we have cells with values greater or equal to the threshold styled in green, and simultaneously other cells with values less than the threshold styled in.... lightgreen?

We absolutely can, we've already done the first part (in the previous code chunk), we just need to add a second condition in a similar manner but in a different `tab_style` statement:

```
t6 <- t5 %>%
  tab_style(
    style = cell_text(
      color = 'lightgreen'
    ),
    location = cells_body(
      columns = 'previous_positive',
      rows = previous_positive < 15700
    )
  )
t6
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Question 1: Conditional Formatting To highlight (in yellow) rows in a `gt` table where the “hiv_positive” column exceeds 1,000, which R code snippet should you use?

A.



```
data %>%
  gt() %>%
  tab_style(
    style = cells_body(),
    columns = "Sales",
    conditions = style_number(Sales > 1000, background =
      "yellow")
  )
```

B.

```
data %>%
  gt() %>%
  tab_style(
    style = cells_data(columns = "Sales"),
    conditions = style_number(Sales > 1000, background =
      "yellow")
  )
```

C.

```
data %>%
  gt() %>%
  tab_style(
    style = cell_fill(
      color = "yellow"
    ),
    locations = cells_body(
      columns = "hiv_positive",
      rows = hiv_positive > 1000
    )
  )
```

PRACTICE



(in RMD)

D.

```
data %>%
  gt() %>%
  tab_style(
    style = cells_data(columns = "Sales"),
    conditions = style_text(Sales > 1000, background =
      "yellow")
  )
```

Question 2: Cell Coloration Fill

Using the `hiv_malawi` data frame, create a `gt` table that displays the total number (**sum**) of “*new_positive*” cases for each “*region*”. Highlight cells with values more than 50 cases in *red* and cells with less or equal to 50 in *green*. Complete the missing parts (_____) of this code to achieve this.

PRACTICE



(in RMD)

```
# Calculate the total_new_pos summary
total_summary <- hiv_malawi %>%
  group_by(_____) %>%
  summarize(total_new_positive = _____)

# Create a gt table and apply cell coloration
summary_table <- total_summary %>%
  gt() %>%
  tab_style(
    style = cell_fill(color = "red"),
    locations = _____(
      columns = "new_positive",
      rows = _____
    )
  ) %>%
  tab_style(
    style = _____,
    locations = cells_body(
      columns = "new_positive",
      _____ new_positive <= 50
    )
  )
)
```

Fonts and text

Now is a good time to add some customization to the text in the table. using the same function `gt::tab_style`.

Let's change the font and color of the title and the subtitle for example, I'm choosing to use the *Yanone Kaffeesatz* font from google. Google fonts provide you with hundreds of thousands of fonts and styles to choose from that can be more interesting than the boring rigid excel fonts.

In order to do that, we need to specify some details in `gt::tab_style` function:

- Argument `style`: assigned the `cell_text` function:
 - We use the `google_font` function to use the chosen font from google and assign it to the argument `font`
 - We specify the color of the text.
- Argument `locations`: assigned the `cells_title` function:
 - We specify the location `title` and `subtitle` through the argument `groups` inside a vector notation `c(...)`

SIDE NOTE

Note that in order to make changes to the appearance of either the title or subtitle, you can simply use `locations = cells_title(groups = "title")` to apply changes to the title, or `locations = cells_title(groups = "subtitle")` to apply changes to the subtitle without the need to use `c(...)`.

```
t7 <- t6 %>%
  tab_style(
    style = cell_text(
      font = google_font(name = 'Yanone Kaffeesatz'),
      color = "pink"
    ),
    locations = cells_title(groups = c("title", "subtitle"))
  )
t7
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Additionally, we can conduct the same changes to the column labels and the rows labels as well, all we need to do is to correctly specify the location of the changes we want to make, except that this time we are changing the background color(or fill color) of the cells we are going to change. We can use that by adding another style function `cell_fill` where we provide the color we want for the background of our cells. lastly, in the `locations` argument, and similar to the `style` argument, we assign a list in which we provide the location information of the changes we want done using the `cells_column_labels` function, where we specify which column labels we want to change.

```

t8 <- t7 %>%
  tab_style(
    style = list(
      cell_text(
        font = google_font(name = "Righteous"),
        color = "pink"
      )
    ),
    locations = list(

      cells_column_labels(columns = everything()), # select every column
      cells_column_spanners(spanners = everything()) # select all spanners
    )
  )
t8

```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

In a similar manner we can do the same thing to the group rows and the periods, all we need to do is add them to locations argument using `cells_rows_groups` for the group rows, and `cells_body` for the rest of the period column as follows:


```
t9 <- t8 %>%
  tab_style(
    style = list( # we wrap the arguments in a list when we have multiple
                  # styling
    cell_text( # text styling
      font = google_font(name = "Righteous"),
      color = "pink"
    ),
    cell_fill(color = "gray") # cell background color fill styling
  ),
  locations = list( # similar with locations
    cells_row_groups(groups = everything()),
    cells_body(columns = period)
  )
)
t9
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

KEY POINT



The idea behind what we've done here is to give you control on what YOU want to achieve and not an example of what you have to do exactly, there's endless ways to customize a `gt` table, it's up to you to choose what you need, and what works for your workflow.

PRACTICE



(in RMD)

Question 2: Fonts and Text Which R code snippet allows you to change the font size of the footnote text in a `gt` table?

A.

```
data %>%
  gt() %>%
  tab_header(font.size = px(16))
```

B.

```
data %>%
  gt() %>%
  tab_style(
    style = cell_text(
      size = 16
    ),
    locations = cells_footnotes()
  )
```

PRACTICE



C.

```
data %>%
  gt() %>%
  tab_style(
    style = cells_header(),
    css = "font-size: 16px;"
  )
```

D.

```
data %>%
  gt() %>%
  tab_style(
    style = cells_header(),
    css = "font-size: 16px;"
  )
```

Borders

In `gt` it's also possible to draw borders in the tables to help the end user focus on specific area in the table. In order to add borders to a `gt` table we will use, again the, `tab_style` function and, again, specify the style and locations argument. The only difference now is that we will use the `cell_borders` helper function and assign it to the style argument. Here's how:

Let's first add a vertical line:

```
t10 <- t9 %>%
  tab_style(
    style = cell_borders( # we are adding a border
      sides = "left",    # to the left of the selected location
      color = "pink",    # with a pink color
      weight = px(5)     # and five pixels of thickness
    ),
    locations = cells_body(columns = 2) # add this border line to the left of
      column 2
  )
t10
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Now let's add another pink horizontal border line:

```
t11 <- t10 %>%
  tab_style(
    style = cell_borders( # we are adding a border line
      sides = "bottom",  # to the bottom of the selected location
      color = "pink",    # with a pink color
      weight = px(5)     # and five pixels of thickness
    ),
    locations = cells_column_labels(columns = everything()) # add this border
      line to the bottom of the column labels
  )
t11
```

HIV Testing in Malawi

Q1 to Q2 2019

	New tests		Previous tests	
	Positive	Negative	Positive	Negative
2019 Q1	6199	284694	14816	6595
2019 Q2	6132	282249	15101	5605
2019 Q3	5907	300529	15799	6491
2019 Q4	5646	291622	15700	6293

Source: Malawi HIV Program

Question 4: Borders To add a solid border around the entire `gt` table, which R code snippet should you use?

Hint : we can use a function that sets options for the entirety of the table, just like the `theme` function for the `ggplot` package.

A.

```
data %>%
  gt() %>%
  tab_options(table.border.top.style = "solid")
```

B.

```
data %>%
  gt() %>%
  tab_options(table.border.style = "solid")
```

C.

```
data %>%
  gt() %>%
  tab_style(
    style = cells_table(),
    css = "border: 1px solid black;"
  )
```

CHALLENGE



D.

CHALLENGE



```
data %>%  
  gt() %>%  
  tab_style(  
    style = cells_body(),  
    css = "border: 1px solid black;"  
  )
```

Answer Key

1.C

2.

```
# Solutions are where the numbered lines are  
  
# Calculate the total_new_pos summary  
total_summary <- hiv_malawi %>%  
  group_by(region) %>% ##1  
  summarize(total_new_positive = new_positive) ##2  
  
# Create a gt table and apply cell coloration  
summary_table <- total_summary %>%  
  gt() %>% ##3  
  tab_style(  
    style = cell_fill(color = "red"),  
    locations = cells_body( ##4  
      columns = "new_positive",  
      rows = new_positive >= 50 ##5  
    )  
  ) %>%  
  tab_style(  
    style = cell_fill(color = "green"), ##6  
    locations = cells_body(  
      columns = "new_positive",  
      rows = new_positive < 50 ##7  
    )  
  )
```

3.B

4.B

Contributors

The following team members contributed to this lesson:



BENNOUR HSIN

Data Science Education Officer
Data Visualization enthusiast



JOY VAZ

R Developer and Instructor, the GRAPH Network
Loves doing science and teaching science

External resources and packages

- The definite cookbook of `gt` by Tom Mock : <https://themockup.blog/static/resources/gt-cookbook.html#introduction>
- the Grammar of Table article : <https://themockup.blog/posts/2020-05-16-gt-a-grammar-of-tables/#add-titles>
- official `gt` documentation page : <https://gt.rstudio.com/articles/intro-creating-gt-tables.html>
- Create Awesome HTML Table with `knitr::kable` and `kableExtra` book by Hao Zhu : https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html#Overview