Améliorer les visualisations de tableaux gt

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

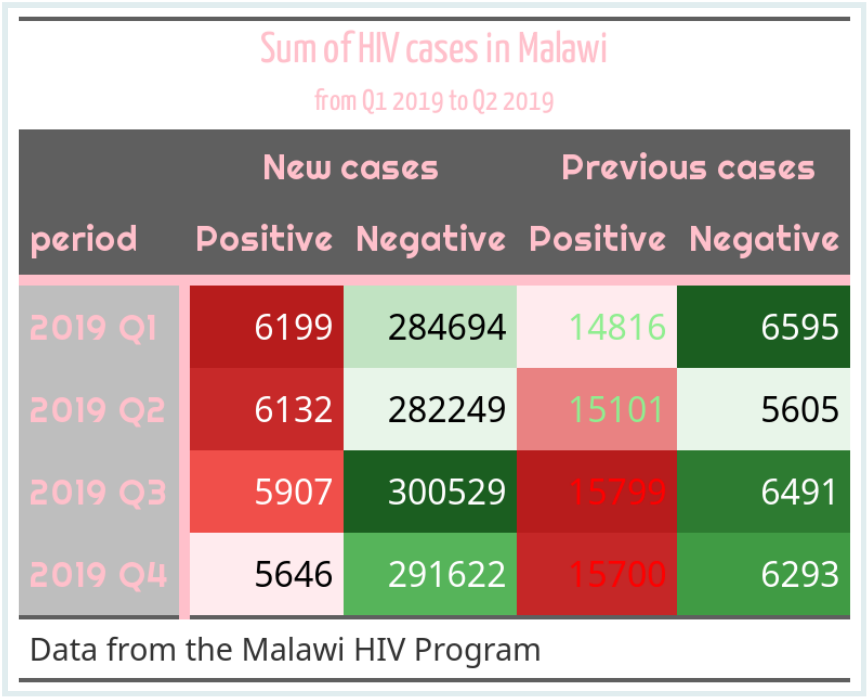
La leçon précédente `gt` s'est principalement concentrée sur les composants de la table, sa structure et comment la manipuler correctement. Cette leçon, présentant la deuxième partie de la série `gt`, se concentrera sur l'utilisation du package pour perfectionner, styliser et personnaliser les effets visuels des tables d'une manière qui élève la qualité et l'efficacité de vos rapports.

Plongeons-y.

Objectifs d'apprentissage

- Formatage des cellules
- Coloration conditionnelle
- Formater le texte (couleur de la police, gras, etc.)
- Ajouter des bordures au texte

À la fin de cette leçon, vous aurez les compétences pour styliser artistiquement vos tables `gt` pour répondre à vos préférences spécifiques, atteignant un niveau de détail similaire à ceci :



Sum of HIV cases in Malawi

from Q1 2019 to Q2 2019

	New cases		Previous cases	
period	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

Dans cette leçon, nous utiliserons les packages suivants :

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

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

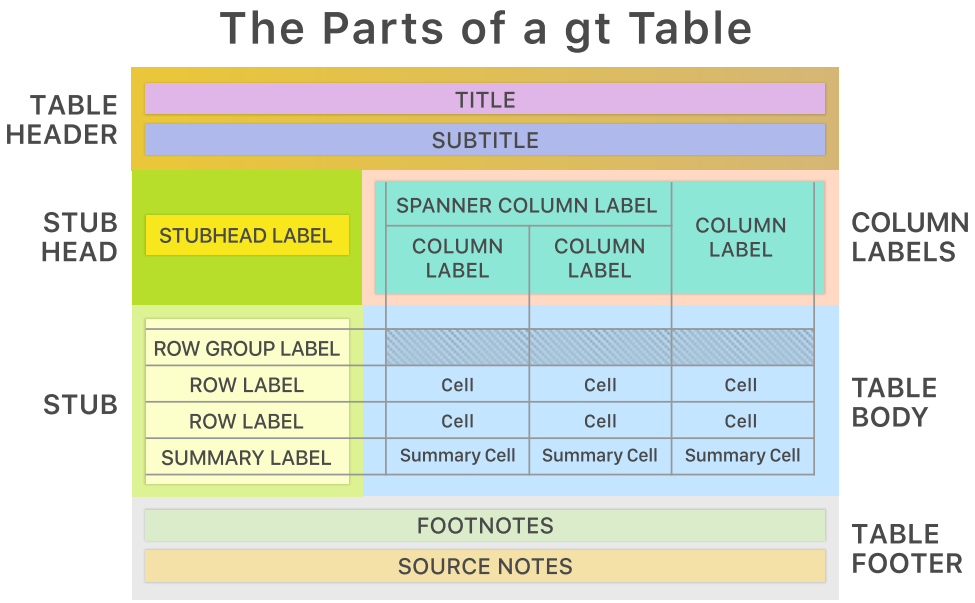
Précédemment dans pt1

RECAP

Dans la leçon `gt` précédente, nous avons eu l'opportunité de :

- Découvrir les données sur la prévalence du VIH au Malawi.
- Découvrir la grammaire des tables et le package `gt`.
- créer une table simple.
- Ajouter des détails comme le titre et la note de bas de page à la table.
- Regrouper les colonnes en en-têtes.
- Créer des lignes de résumé.

RECAP



Jeu de données

Dans cette leçon, nous utiliserons les mêmes données que la leçon précédente. Vous pouvez revenir en arrière pour une description détaillée des données et du processus de préparation que nous avons effectué.

RECAP

Voici les détails complets des colonnes que nous utiliserons :

RECAP

- region: La région géographique ou la zone où les données ont été collectées ou sont analysées.
- period: Une période spécifique associée aux données, souvent utilisée pour l'analyse temporelle.
- previous_negative: Le nombre d'individus avec un résultat de test négatif précédent.
- previous_positive: Le nombre d'individus avec un résultat de test positif précédent.
- new_negative: Le nombre de nouveaux cas diagnostiqués avec un résultat négatif.
- new_positive: Le nombre de nouveaux cas diagnostiqués avec un résultat positif.

Mais pour les besoins de cette leçon, nous utiliserons directement les tableaux. Voici le tableau que nous avons créé avec les bonnes en-têtes et étiquettes de colonnes. Nous baserons le reste de notre leçon sur ce tableau en particulier.

```
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

Thèmes

Comme l'objectif de cette leçon est principalement le style, commençons par utiliser un thème prédéfini pour ajouter plus de visuels et de couleurs au tableau et à ses composants. Pour ce faire, nous utilisons la fonction `opt_stylize`. La fonction contient plusieurs styles prédéfinis et peut accepter une couleur également. Dans notre cas, nous avons choisi d'opter pour le style No.6 et la couleur 'cyan'. Vous pouvez définir ces arguments selon vos préférences.

```
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

Pour des thèmes et des styles plus sophistiqués, vous pouvez consulter la fonction `tab_options` (documentation ici) qui est essentiellement l'équivalent de la fonction `theme` dans `ggplot2`. Cette fonction contient des arguments et des options pour chaque couche et composant du tableau. Pour les besoins de cette leçon, nous n'approfondirons pas.

Formatage des valeurs dans le tableau

Ne serait-il pas utile de visualiser en couleurs la différence entre les valeurs d'une colonne spécifique ? Dans de nombreux rapports, ces types de tableaux sont très utiles, surtout si le nombre de lignes est assez grand. Faisons cela pour notre tableau de sorte que la colonne `new_positive` soit formatée en rouge.

```
# révisé la documentation, paletteer n'est pas nécessaire.
# my_colors <- paletteer::paletteer_d("ggsci::red_material")
```

Nous pouvons le faire à l'aide de la fonction `data_color` pour laquelle nous devons spécifier deux arguments : `columns` (c'est-à-dire sur quelle colonne ce style sera-t-il appliqué ?) et `palette` pour la palette de couleurs que nous souhaitons utiliser.

```
t2 <- t1 %>%
  data_color(
    columns = new_positive, # la colonne ou les colonnes comme nous le verrons
        plus tard
    palette = "ggsci::red_material" # la palette du package ggsci.
  )

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` n'est pas la seule palette que nous pouvons utiliser, en fait, il existe des centaines de palettes conçues pour être utilisées dans R. Vous pouvez en trouver beaucoup plus dans la

documentation du package `paletteer`ici, ou dans la documentation officielle de `data_color`ici.

Nous pouvons faire cela pour la colonne `previous_negative` également. Nous pouvons utiliser un autre type de palette, pour ce cas, j'utilise la palette verte du même package : `ggsci::green_material`, la palette que vous choisissez est une question de commodité et de goût personnel, vous pouvez en savoir plus à ce sujet si vous vous référez à la note de marge ci-dessus.

```
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

De même, nous pouvons également colorer plusieurs colonnes à la fois, par exemple nous pouvons styliser les colonnes avec des cas positifs en rouge, et celles avec des cas négatifs en vert. Pour faire cela, nous devons écrire *deux* instructions `data_color`, une pour chaque style de couleur :

```r
t4 <- t1 %>%
  data_color(
    columns = ends_with("positive"), # sélection des colonnes se terminant par
        le mot positive
    palette = "ggsci::red_material" # palette rouge
  ) %>%
  data_color(
    columns = ends_with("negative"), # sélection des colonnes se terminant par
        le mot negative
    palette = "ggsci::green_material" # palette verte
  )

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

Rappelez-vous, dans la leçon précédente, nous avons utilisé les fonctions `tidyselect` pour sélectionner des colonnes, dans le code ci-dessus, nous avons utilisé la fonction `ends_with` pour sélectionner les colonnes se terminant soit par le mot 'négatif' soit par 'positif', ce qui est parfait pour l'objectif de notre tableau.

De plus, les étiquettes des colonnes dans le tableau `gt` et les noms réels des colonnes dans le `data.frame` peuvent être différents, dans notre cas, nous nous référons aux noms dans les données.

Formatage conditionnel

Nous pouvons également configurer le tableau pour changer conditionnellement le style d'une cellule en fonction de sa valeur. Dans notre cas, nous voulons mettre en évidence les valeurs de la colonne `previous_positive` en fonction d'un seuil (la valeur 15700). Les valeurs supérieures ou égales au seuil devraient être en vert.

Pour ce faire, nous utilisons la fonction `tab_style` où nous spécifions deux arguments :

- `style` : où nous spécifions la couleur dans la fonction `cell_text` car nous avons l'intention de manipuler le texte à l'intérieur des cellules.
- `locations` : où nous spécifions les colonnes et les rangées de notre manipulation dans `cells_body` puisque ces cellules sont dans le corps principal du tableau.

```
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

Dans le code ci-dessus, la condition sur laquelle le style s'appliquera est déclarée dans :

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

Notez également que nous pouvons passer plus d'arguments à la fonction `cell_text`, comme la taille et la police des cellules que nous souhaitons styler.

Et si nous voulions avoir une condition à deux côtés sur le même seuil ? Pouvons-nous avoir des cellules avec des valeurs supérieures ou égales au seuil stylisées en vert, et simultanément d'autres cellules avec des valeurs inférieures au seuil stylisées en…. lightgreen ?

Nous le pouvons absolument, nous avons déjà fait la première partie (dans le bloc de code précédent), nous devons simplement ajouter une seconde condition de manière similaire mais dans une déclaration `tab_style` différente :

```
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 : Mise en forme conditionnelle Pour mettre en surbrillance (en jaune) les lignes d'un tableau `gt` où la colonne "hiv_positive" dépasse 1 000, quel extrait de code R devriez-vous utiliser ?

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
    )
  )
```

D.

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

Question 2 : Coloration de la cellule

En utilisant la dataframe `hiv_malawi`, créez un tableau `gt` qui affiche le nombre total (**somme**) de cas "*new_positive*" pour chaque "*region*". Mettez en surbrillance les

cellules avec des valeurs supérieures à 50 cas en *rouge* et les cellules avec 50 ou moins en *vert*. Complétez les parties manquantes (_____) de ce code pour y parvenir.

```
# 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
    )
  )
```

Polices et texte

C'est maintenant le bon moment pour ajouter une personnalisation au texte dans le tableau. en utilisant la même fonction `gt::tab_style`.

Changeons la police et la couleur du titre et du sous-titre par exemple, je choisis d'utiliser la police `Yanone Kaffeesatz` de google. Google fonts vous fournit des centaines de milliers de polices et de styles parmi lesquels choisir, qui peuvent être plus intéressants que les polices Excel rigides et ennuyeuses.

Pour cela, nous devons spécifier quelques détails dans la fonction `gt::tab_style` :

- Argument `style` : attribue la fonction `cell_text` :
    - Nous utilisons la fonction `google_font` pour utiliser la police choisie depuis google et l'attribuer à l'argument `font`
    - Nous spécifions la couleur du texte.
- Argument `locations` : attribue la fonction `cells_title` :
    - Nous spécifions l'emplacement `title` et `subtitle` via l'argument `groups` dans une notation vectorielle `c(...)`

Notez que pour apporter des modifications à l'apparence du titre ou du sous-titre, vous pouvez simplement utiliser : `locations = cells_title(groups = "title")` pour appliquer des modifications au titre, ou `locations = cells_title(groups = "subtitle")` pour appliquer des modifications au sous-titre sans avoir besoin d'utiliser `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

De plus, nous pouvons effectuer les mêmes changements sur les libellés de colonnes et de lignes, tout ce que nous devons faire est de spécifier correctement l'emplacement des modifications que nous souhaitons apporter, sauf que cette fois nous modifions la couleur d'arrière-plan (ou couleur de remplissage) des cellules que nous allons modifier. Nous pouvons le faire en ajoutant une autre fonction de style `cell_fill` où nous fournissons la couleur que nous voulons pour l'arrière-plan de nos cellules. Enfin, dans l'argument `locations`, et de manière similaire à l'argument `style`, nous attribuons une liste dans laquelle nous fournissons les informations de localisation des modifications que nous souhaitons apporter à l'aide de la fonction `cells_column_labels`, où nous spécifions les libellés de colonne que nous souhaitons modifier.

```r
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

De manière similaire, nous pouvons faire la même chose pour les lignes de groupe et les périodes, tout ce que nous devons faire est de les ajouter à l'argument locations en utilisant `cells_rows_groups` pour les lignes de groupe, et `cells_body` pour le reste de la colonne période comme suit :

```r
t9 <- t8 %>%
  tab_style(
    style = list( # we wrap the arguments in a list when we have multiple
        stuling
      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

L'idée derrière ce que nous avons fait ici est de vous donner le contrôle sur ce que VOUS voulez réaliser et non un exemple de ce que vous devez faire exactement. Il y a d'innombrables façons de personnaliser une table `gt`. C'est à vous de choisir ce dont vous avez besoin et ce qui fonctionne pour votre flux de travail.

Question 2: Polices et Texte Quel extrait de code R permet de changer la taille de la police du texte de bas de page dans une table `gt` ?

A.

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

B.

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

C.

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

D.

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

Bordures

Avec `gt`, il est également possible de dessiner des bordures dans les tables pour aider l'utilisateur final à se concentrer sur une zone spécifique de la table. Pour ajouter des bordures à une table `gt`, nous utiliserons, encore une fois, la fonction `tab_style` et, de nouveau, spécifierons l'argument style et locations. La seule différence maintenant est que nous utiliserons la fonction d'aide `cell_borders` et l'attribuerons à l'argument style. Voici comment :

Commençons par ajouter une ligne verticale :

```
t10 <- t9 %>%
  tab_style(
    style = cell_borders( # nous ajoutons une bordure
      sides = "left",     # à gauche de l'emplacement sélectionné
      color = "pink",     # avec une couleur rose
      weight = px(5)      # et cinq pixels d'épaisseur
    ),
    locations = cells_body(columns = 2) # ajoutez cette ligne de bordure à
        gauche de la colonne 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

Ajoutons maintenant une autre ligne de bordure horizontale rose :

```
t11 <- t10 %>%
    tab_style(
    style = cell_borders( # nous ajoutons une ligne de bordure
      sides = "bottom",   # en bas de l'emplacement sélectionné
      color = "pink",     # avec une couleur rose
      weight = px(5)      # et cinq pixels d'épaisseur
    ),
    locations = cells_column_labels(columns = everything()) # ajoutez cette
        ligne de bordure en bas des étiquettes de colonne
  )

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

CHALLENGE

Question 4 : Bordures Pour ajouter une bordure solide autour de l'ensemble du tableau `gt`, quel extrait de code R devriez-vous utiliser ?

Astuce : nous pouvons utiliser une fonction qui définit des options pour l'ensemble du tableau, tout comme la fonction `theme` pour le package `ggplot`.

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.

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

Corrigé

1.C

2.

```
# Les solutions sont là où les lignes sont numérotées

# Calculez le résumé total_new_pos
total_summary <- hiv_malawi %>%
  group_by(region) %>% ##1
  summarize(total_new_positive = new_positive) ##2

# Créez un tableau gt et appliquez une coloration de cellule
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

Contributeurs

Les membres suivants de l'équipe ont contribué à cette leçon :

BENNOUR HSIN

Data Science Education Officer
Data Visualization enthusiast

JOY VAZ

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

Ressources et packages externes

- 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