

Pyramides démographiques pour l'analyse épidémiologique

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
Objectifs d'Apprentissage
Introduction aux pyramides démographiques
Utilisation des pyramides de population en épidémiologie
Conceptualisation des pyramides démographiques
Packages
Préparation des Données
Introduction à l'ensemble de données
Importation des Données
Création d'un Sous-ensemble de Données Agrégées
Création de graphiques
Utilisation de <code>geom_col</code>
Personnalisation de graphique
Etiquettes
Axe
Schéma de Couleurs et Thèmes
Récap !
Corrigé
Références

Introduction

Aujourd'hui, vous allez apprendre l'importance d'utiliser des pyramides démographiques pour visualiser la distribution d'une maladie par âge et sexe et comment en créer une avec `ggplot2`.

Commençons !

Objectifs d'Apprentissage

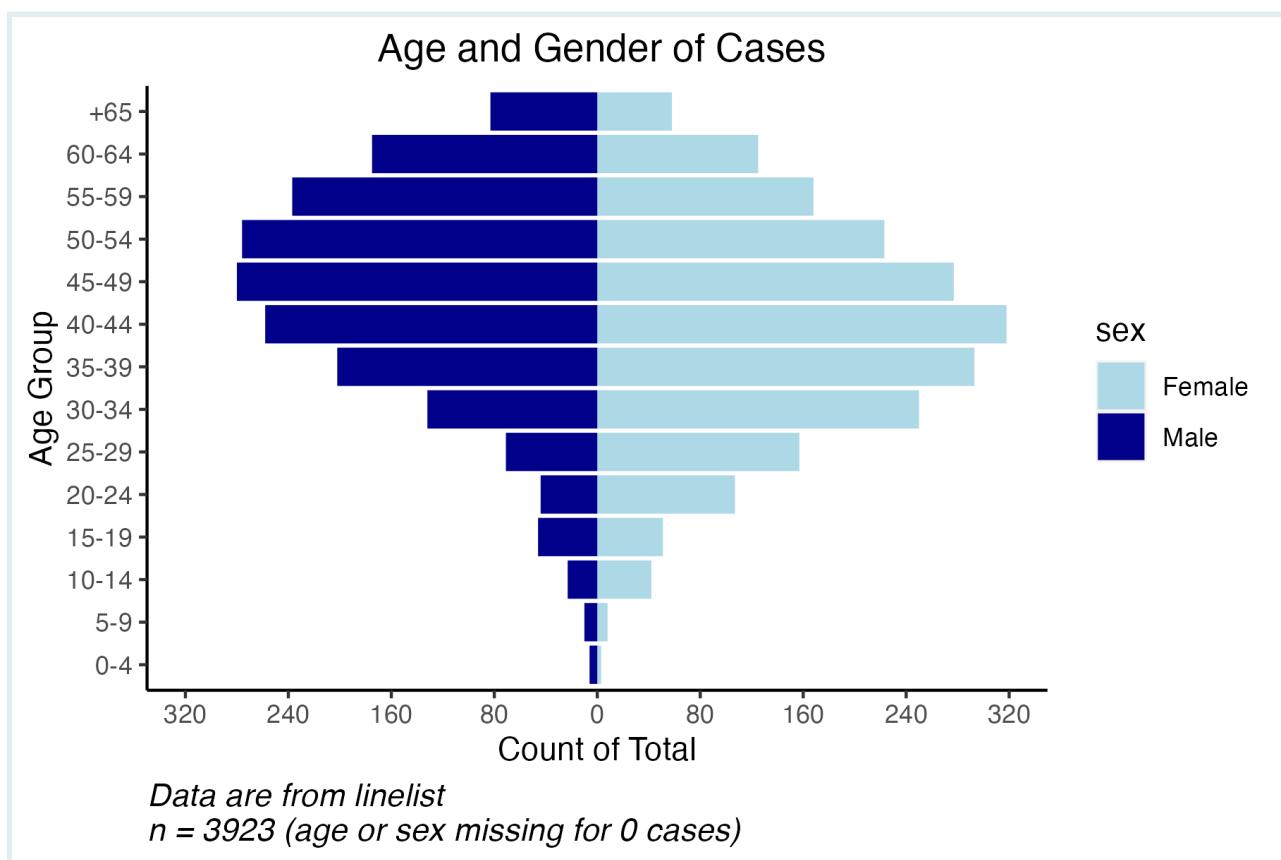
- Vous connaissez l'importance d'utiliser des pyramides démographiques pour montrer la distribution des groupes d'âge et du sexe pour les maladies transmissibles.
- Vous pouvez utiliser `geom_col` de `ggplot2` pour créer une pyramide démographique montrant le nombre ou le pourcentage total de cas, décès, et plus encore.
- Vous pouvez personnaliser le graphique en changeant le schéma de couleurs, les étiquettes et les axes.

Introduction aux pyramides démographiques

Une pyramide démographique, également connue sous le nom de pyramide de population ou pyramide âge sexe, aide à visualiser la distribution d'une population selon deux variables démographiques importantes : **l'âge** et le **sexé**. La population globale prend la forme d'une pyramide, d'où son nom.

Les pyramides de population sont des graphiques qui montrent la distribution des âges dans une population divisée au centre entre les membres masculins et féminins de la population, où l'axe des ordonnées montre les groupes d'âge et l'axe des abscisses le sexe.

Grâce à l'utilisation de `ggplot2`, nous sommes en mesure de créer des pyramides tout en les personnalisant selon nos besoins spécifiques, comme le graphique ci-dessous :



Plusieurs packages sont disponibles pour faciliter l'analyse des données et la visualisation des données. Dans le cas des pyramides démographiques, le package `apyparid` peut être un outil utile. Néanmoins, en utilisant ce package, nous sommes limités dans la personnalisation de notre graphique, ce qui rend `ggplot2` une approche beaucoup plus polyvalente.

SIDE NOTE



SIDE NOTE

Le package `apyramid` est un produit du projet **R4Epis** qui permet la création rapide de pyramides de population et de nombreuses autres fonctions utiles utilisées dans les rapports épidémiologiques.

Des informations détaillées sur le package peuvent être lues [ici](#) ou en entrant `?age_pyramid` dans votre console R.

Utilisation des pyramides de population en épidémiologie

Pour décrire et comprendre l'épidémiologie de diverses maladies transmissibles, les pyramides de population fournissent des informations utiles tout en facilitant la visualisation de la distribution de la maladie par âge et par sexe.

Nous savons que l'incidence de certaines maladies transmissibles peut varier avec l'âge. Dans le cas de la tuberculose (TB), les adolescents et les jeunes adultes sont principalement touchés dans la région d'Afrique. Cependant, dans les pays où la TB est passée d'une forte à une faible incidence, comme les États-Unis, la TB est principalement une maladie des personnes âgées ou immunodéprimées. Une autre maladie qui démontre une variation en fonction de l'âge est le paludisme, où les enfants de moins de 5 ans représentent une grande majorité des décès dans la région d'Afrique.

Par conséquent, lors de la description de l'épidémiologie de maladies transmissibles telles que le VIH, le paludisme et la tuberculose, il est important d'observer la distribution des cas ou des décès par groupe d'âge et par sexe. Ces informations aident à informer les programmes de surveillance nationaux sur le groupe d'âge qui connaît le fardeau le plus lourd et celui à cibler pour l'intervention.

Utilisation de la distribution démographique pour l'évaluation de la qualité des données

SIDE NOTE

Les pyramides démographiques peuvent également jouer un rôle crucial dans l'évaluation de la qualité des données des systèmes de surveillance de routine en aidant à évaluer la cohérence interne et externe.

Lorsqu'on essaie d'évaluer les normes de qualité des données de surveillance de certaines maladies, la cohérence externe peut être évaluée en comparant les données de surveillance nationales à l'épidémiologie mondiale de cette maladie. Les calculs de données basés sur des variables démographiques comme le groupe d'âge sont parfois utilisés.

Dans le cas des données de surveillance de la tuberculose, la cohérence externe peut être évaluée en calculant le pourcentage d'enfants

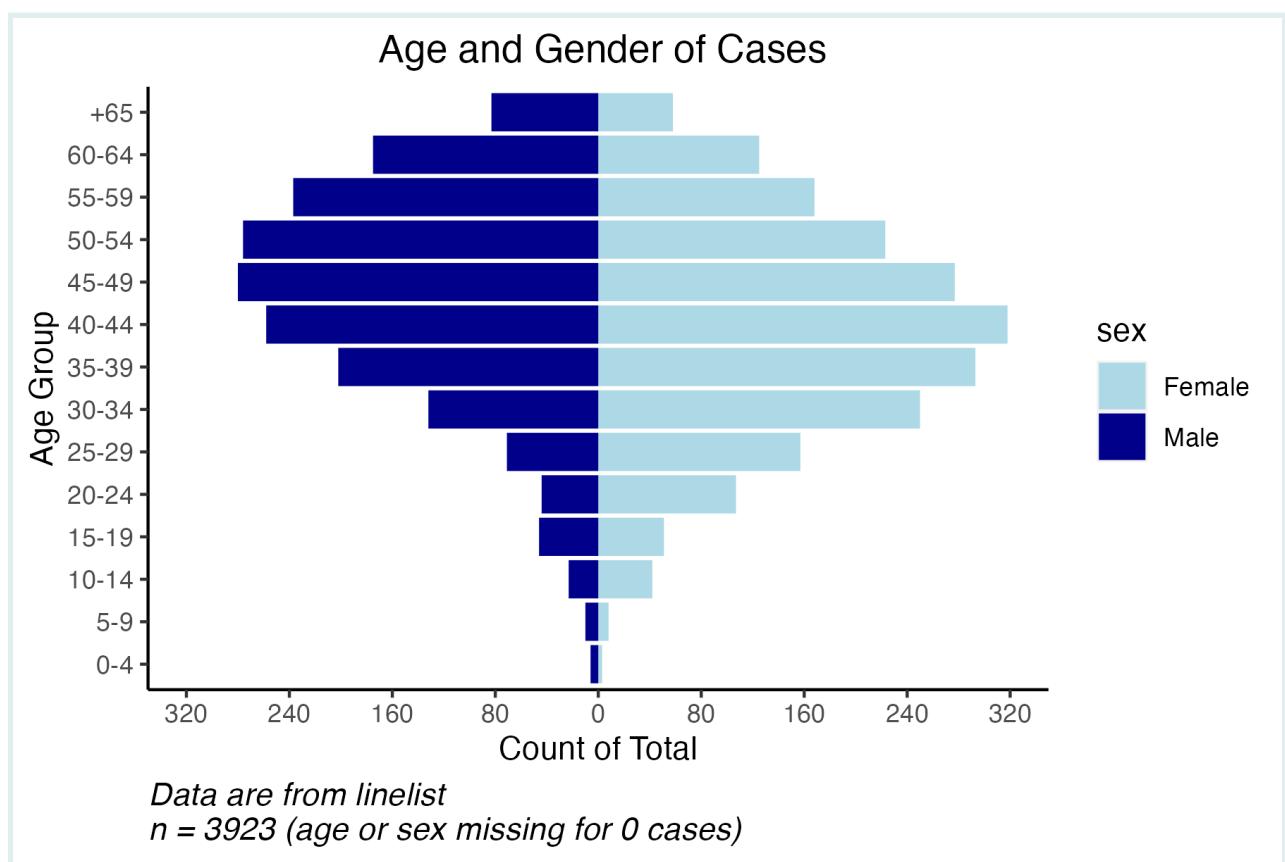
SIDE NOTE

diagnostiqués tuberculeux dans le programme et en le comparant à la moyenne mondiale des cas.

Conceptualisation des pyramides démographiques

Prenons un regard plus approfondi sur une pyramide de population et essayons de comprendre comment elle peut être représentée graphiquement à l'aide de `geom_col` de `ggplot2`.

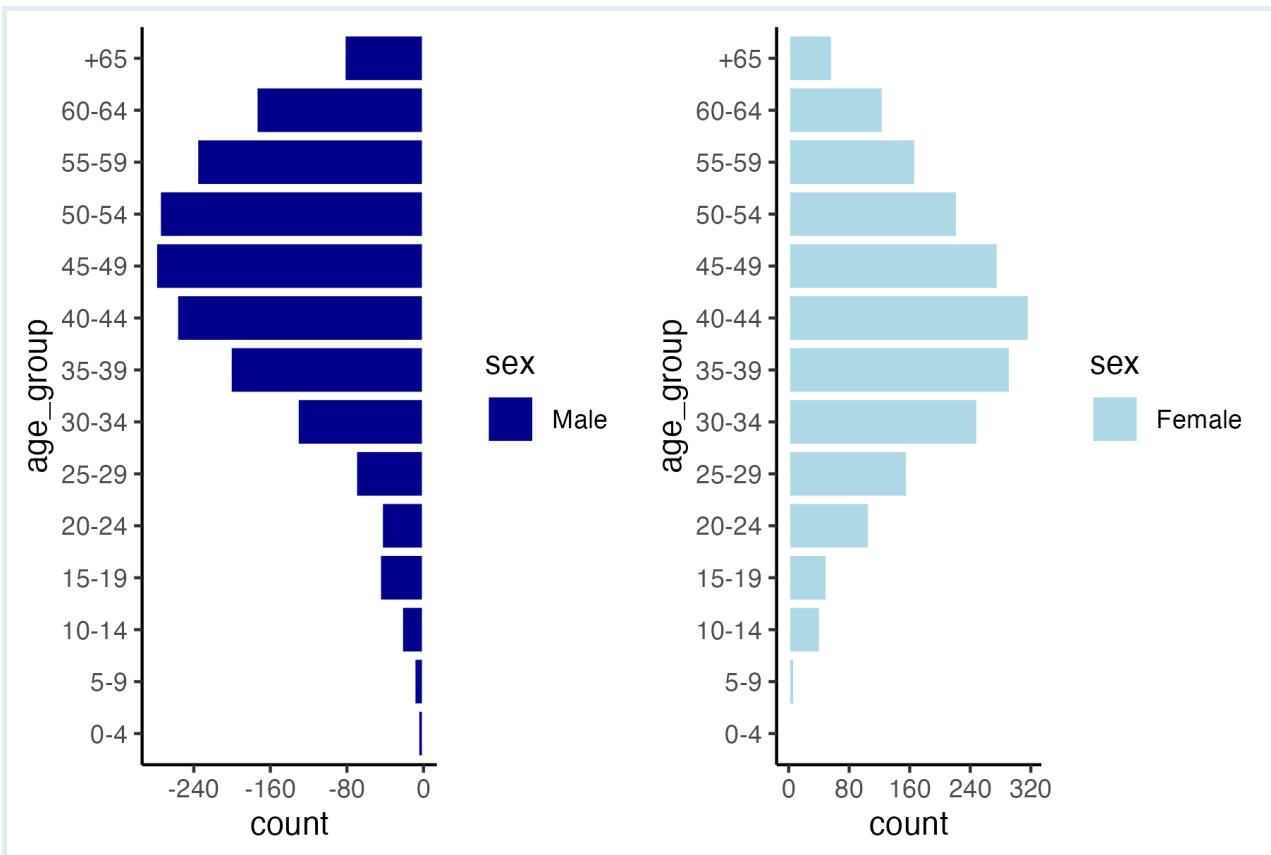
Pour conceptualiser cela, nous utiliserons l'ensemble de données introduit plus tard dans la leçon. Regardons de plus près la pyramide démographique que nous avons montrée ci-dessus:



Comme vous pouvez le voir sur l'image ci-dessus, l'axe des x est divisé en deux moitiés (hommes et femmes) où les unités de l'axe des x sont symétriques de part et d'autre de l'axe au point 0 et les groupes d'âge sont étiquetés le long de l'axe des y.

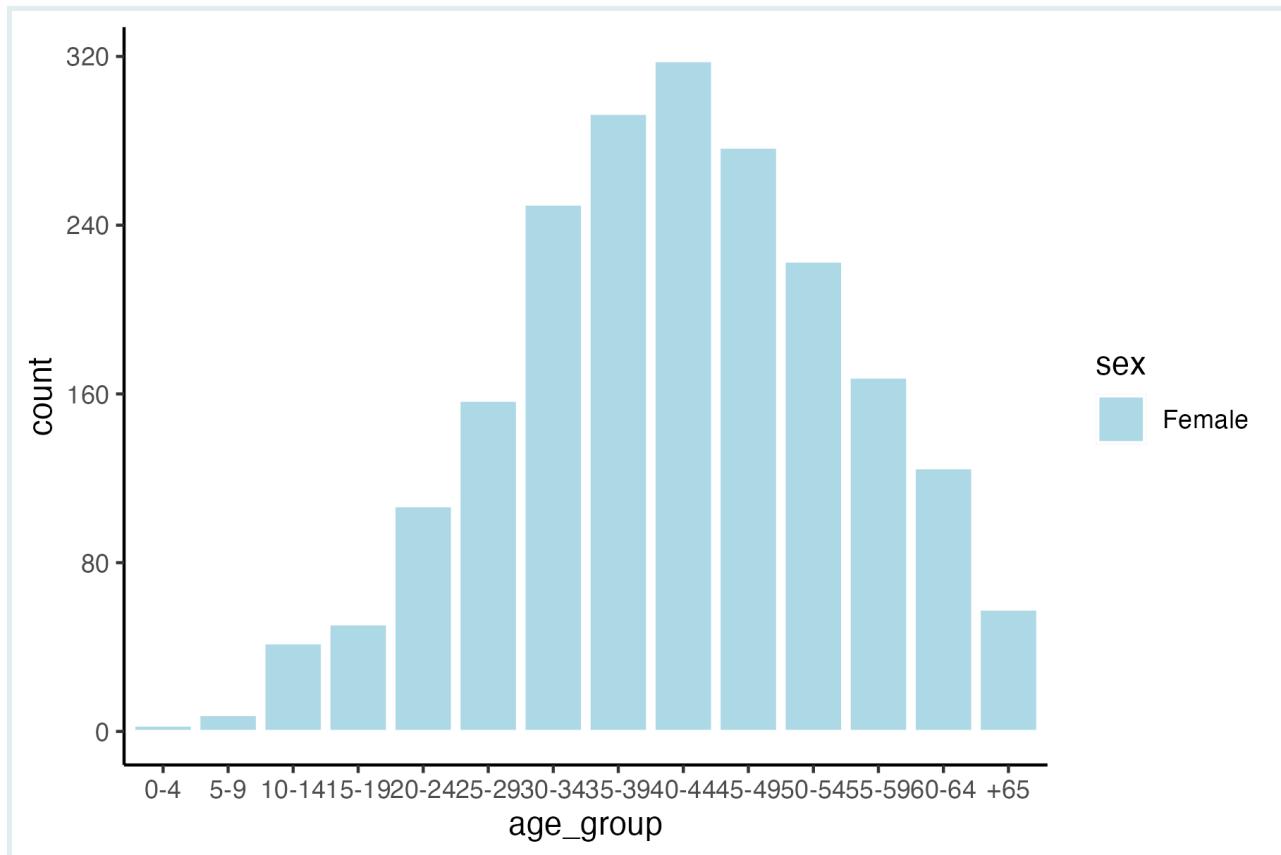
En regardant de près ce graphique, vous avez probablement remarqué que la pyramide de population est constituée de deux tracés fusionnés à travers l'axe des y.

En d'autres termes, nous pouvons diviser la pyramide en deux sections (hommes ou femmes) et les représenter graphiquement de manière indépendante comme ci-dessous:



Ce type de graphique vous semble-t-il familier ? Et si nous inversions nos axes x et y et tournions le graphique de 90 degrés ?

Prenons la moitié féminine et retournons nos axes pour voir ce que nous obtenons :



Comme vous pouvez probablement le constater maintenant, pour créer une pyramide de population, nous avons créé deux **graphiques à barres** montrant la répartition des sexes par groupes d'âge. Une fois que nous avons créé un graphique pour les femmes et les hommes, nous les avons ensuite fusionnés ensemble à travers leur axe des y.



Pour que la moitié masculine soit représentée à gauche de l'axe des y, nous devrons nier le **compte** (nombre total de personnes par groupe d'âge).

En d'autres termes, les pyramides de population sont des **graphiques à barres** dont les axes sont **inversés** (axes x et y inversés) et où les femmes sont représentées du **côté positif** de l'axe des y et les hommes sont représentés du **côté négatif** de l'axe des y.

Packages

Cette leçon nécessitera l'installation et le chargement des "packages" suivants :

```
# Charger les packages
if(!require(pacman)) install.packages("pacman")
pacman::p_load(here,          # pour localiser le fichier
               tidyverse,    # pour nettoyer, gérer et tracer les données
               (inclus le paquet ggplot2)
               janitor,     # tables et nettoyage des données
               apyramid)    # paquet dédié à la création de pyramides des âges
```

Préparation des Données

Introduction à l'ensemble de données

Pour cette leçon, nous utiliserons un ensemble de données fictif sur le VIH imitant une liste linéaire de cas de VIH au Zimbabwe en 2016.



SIDE NOTE

Vous pouvez accéder à la source utilisée pour simuler les données sur le VIH [ici](#)

Chaque ligne (rangée) correspond à un patient, tandis que chaque colonne représente différentes variables d'intérêt. La liste linéaire ne contient que des variables démographiques et liées au VIH (statut VIH).

Pour cette leçon spécifique, nous nous concentrerons sur les variables **liées à l'âge** et **au sexe** pour créer notre pyramide démographique.

Importation des Données

Commençons par importer nos données dans notre environnement RStudio et examinons-les de plus près pour mieux comprendre les variables que nous utiliserons pour la création de notre pyramide démographique.



SIDE NOTE

Afin de concentrer notre attention sur la création de pyramides démographiques, nous avons déjà créé le sous-ensemble de données contenant les variables d'intérêt (`age_group` et `sex`).

```
hiv_data <- read_csv(here::here("data/clean/hiv_zimbabwe_2016.csv"))

hiv_data
```

```

## # A tibble: 10 × 3
##   age_group sex    hiv_status
##   <chr>     <chr>  <chr>
## 1 20-24     female positive
## 2 35-39     male   positive
## 3 15-19     female negative
## 4 40-44     male   negative
## 5 45-49     male   positive
## 6 35-39     female negative
## 7 50-54     male   positive
## 8 20-24     female positive
## 9 5-9       male   negative
## 10 60-64    male  negative

```

Notre ensemble de données importé contient **28000** rangées et **3** colonnes contenant les variables `age_group` et `sex` que nous utiliserons pour la création de notre pyramide démographique. De plus, la variable `hiv_status` nous fournit des informations sur le statut des patients (*positif* ou *négatif*).

Comme nous sommes intéressés à créer une pyramide démographique sur la prévalence du VIH, nous devons d'abord filtrer les patients séropositifs et nous assurer que les variables `age_group` et `hiv_status` sont factorisées.

Chargeons un sous-ensemble de nos données déjà nettoyées !

```

hiv_prevalence <-
  readRDS(here::here("data/clean/hiv_zimbabwe_prevalence.rds"))

hiv_prevalence

```

```

## # A tibble: 10 × 3
##   age_group sex    hiv_status
##   <fct>     <fct>  <fct>
## 1 20-24     female positive
## 2 35-39     male   positive
## 3 45-49     male   positive
## 4 50-54     male   positive
## 5 20-24     female positive
## 6 45-49     female positive
## 7 25-29     male   positive
## 8 35-39     female positive
## 9 60-64     male   positive
## 10 45-49    male  positive

```

SIDE NOTE



Remarquez que nous avons maintenant un sous-ensemble de données de **3923** rangées et **3** colonnes où tous les patients sont **séropositifs** !

Maintenant, avant de passer à la création de notre pyramide démographique, inspectons les données en créant un tableau résumant les colonnes `age_group` et `sex` !

Pour cette étape, nous utiliserons `tabyl()` de `janitor`.

```
hiv_prevalence %>%
  tabyl(age_group, sex)
```

```
##   age_group female male
##       0-4      3     6
##       5-9      8    10
##      10-14     42    23
##      15-19     51    46
##      20-24    107    44
##      25-29    157    71
##      30-34    250   132
##      35-39    293   202
##      40-44    318   258
##      45-49    277   280
```

Selon le tableau, les données sont propres tandis que la colonne `age_group` est correctement organisée par ordre croissant (du plus jeune au plus âgé).



Avant de créer votre pyramide démographique, assurez-vous de vérifier que vos données sont propres et correctement organisées dans un **ordre croissant** ! C'est important lorsque vous utilisez des variables catégorielles car l'ordre de votre `age_group` affectera l'ordre dans lequel il sera tracé dans votre pyramide.

Dans le cas des pyramides démographiques, nous voulons que le groupe d'âge le plus jeune soit situé en bas de l'axe des y et que le groupe d'âge le plus âgé soit en haut de l'axe des y.

Création d'un Sous-ensemble de Données Agrégées

Avant de commencer, nous devrons créer un sous-ensemble de données agrégées qui regroupe le nombre total d'occurrences par groupe d'âge et par sexe comme ci-dessous :

	age_group	sex	count
	<fct>	<fct>	<dbl>
1	0-4	female	3
2	0-4	male	-6
3	5-9	female	8
4	5-9	male	-10
5	10-14	female	42
6	10-14	male	-23
7	15-19	female	51
8	15-19	male	-46
9	20-24	female	107
10	20-24	male	-44
	# i 18 more rows		

Notez que les valeurs *male* sont négatives afin d'obtenir le graphique en barres masculin sur le *côté gauche* du graphique!

Il est important de comprendre comment utiliser `geom_col`.

KEY POINT



Lors de l'utilisation de la fonction `geom_col`, le nombre pour chaque groupe **doit être spécifié dans le `aes` comme variable x ou y**. En d'autres termes, vous devrez utiliser un ensemble de données avec le nombre agrégé d'occurrences pour chaque niveau catégoriel.

Dans le cas des pyramides démographiques, nous devrons utiliser un ensemble de données avec des comptages ou des pourcentages agrégés par groupe d'âge et par sexe.

Commençons par calculer le nombre total et les pourcentages par groupe d'âge et par sexe et créons un sous-ensemble avec ces informations où les valeurs féminines sont *positives* et les valeurs masculines sont *négatives* !

REMINDER



N'oubliez pas de nier la **valeur y masculine** afin d'obtenir le graphique en barres masculin sur le *côté gauche* du graphique!

```

# Créez un nouveau sous-ensemble
pyramid_data <-
  hiv_prevalence %>%
    ungroup() %>%
    count(age_group,
          sex,
          name = "total") %>%
    mutate(
      counts = case_when(
        sex == "female" ~ total,
        sex == "male" ~ -total, #convertit male en négatif
        TRUE ~ NA_real_),
      percent = round(100*(total / sum(total, na.rm = T)), digits = 1),
      percent = case_when(
        sex == "female" ~ percent,
        sex == "male" ~ -percent, #convertit male en négatif
        TRUE ~ NA_real_)) #Rend les valeurs NA numériques également
pyramid_data

```

```

## # A tibble: 5 × 5
##   age_group sex     total counts percent
##   <fct>     <fct>   <int>  <dbl>   <dbl>
## 1 0-4       female    3      3     0.1
## 2 0-4       male      6     -6    -0.2
## 3 5-9       female    8      8     0.2
## 4 5-9       male     10    -10    -0.3
## 5 10-14     female   42     42     1.1

```

SIDE NOTE



Notez que les valeurs masculines dans les colonnes `counts` et `percent` ne sont pas négatives !

Testons votre compréhension avec les questions à choix multiples suivantes (La clé de réponse est située à la fin) :

1. Lors de la création du graphique en barres masculin, quelle modification est apportée aux valeurs d'occurrence?

- a. Elles sont ajoutées aux valeurs de l'axe x.
- b. Elles sont multipliées par 2.

- c. Elles sont divisées par 2.
- d. Elles sont niées (multipliées par -1).

Création de graphiques

Utilisation de `geom_col`

Comme nous allons créer la pyramide démographique en traçant un **graphique à barres**, nous aurons besoin d'utiliser une **variable catégorielle**. Par conséquent, nous utiliserons la variable catégorielle `age_group` pour tracer notre graphique à barres et 'fill' par `sex`.

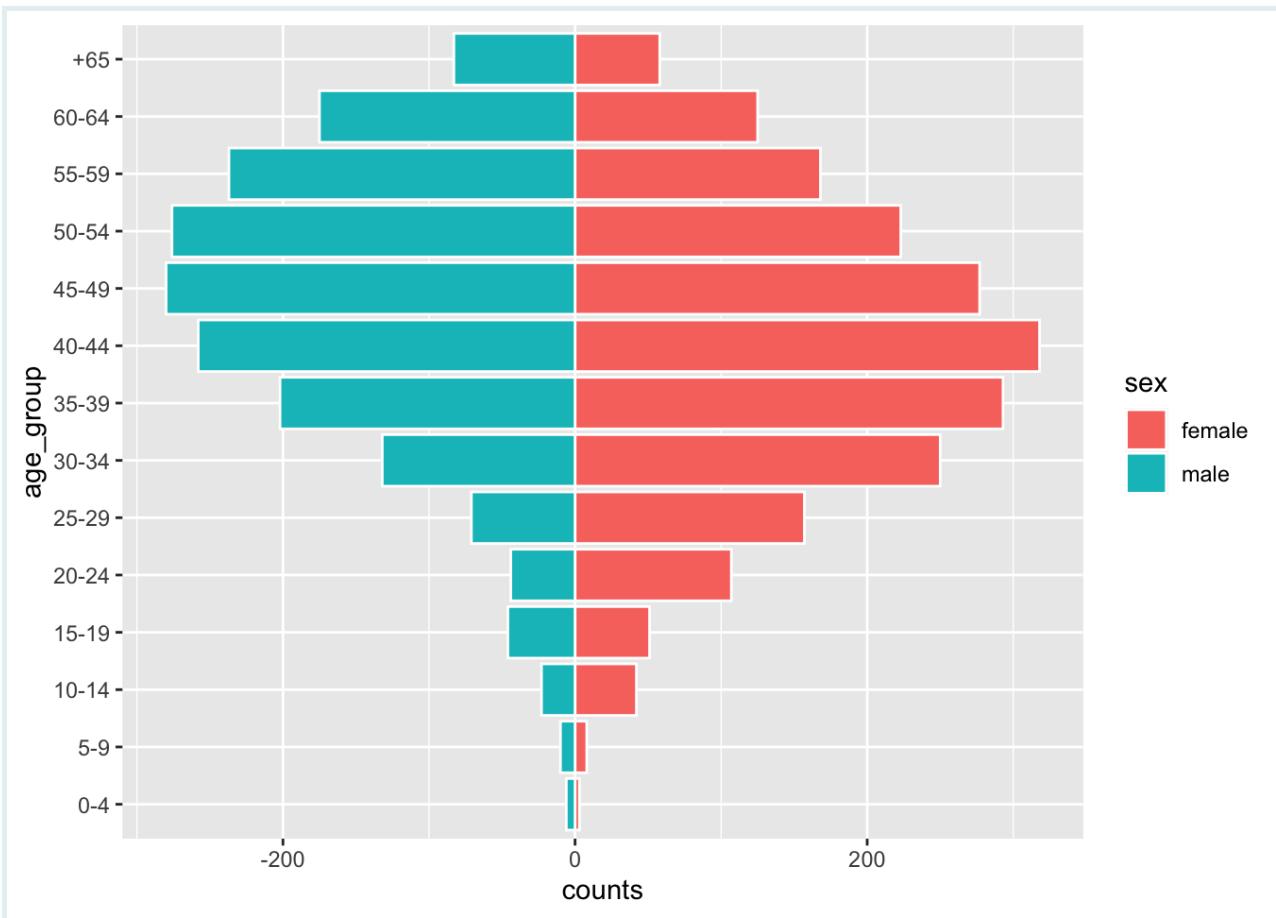
REMINDER



Pour utiliser la fonction `geom_col`, votre jeu de données doit inclure le comptage agrégé ou le pourcentage pour chaque groupe (**groupe d'âge** et **sexe**) !

Créons notre pyramide démographique en utilisant les fonctions `geom_col` de `ggplot2`.

```
demo_pyramid <-
  # Commencer ggplot
  ggplot() +
    # Créer un graphique à barres avec geom_bar
    geom_col(data = pyramid_data, #spécifier les données à tracer
              mapping = aes(
                x = age_group,      #indiquer la variable x
                y = counts,         #indiquer la variable y
                fill = sex),       #remplir par sexe
                colour = "white") +
    # Inverser les axes X et Y
    coord_flip()
demo_pyramid
```



Avez-vous remarqué que nous avons tracé les données et les informations de mappage de notre graphique dans la fonction `geom_col()` au lieu de la fonction `ggplot()` ?

En le traçant directement dans la fonction `geom_col()`, nous sommes en mesure d'ajouter différentes couches à notre graphique (comme des graphiques à barres supplémentaires, des lignes, des points, et plus encore) sans affecter le résultat de notre graphique à barres déjà créé et individuel.

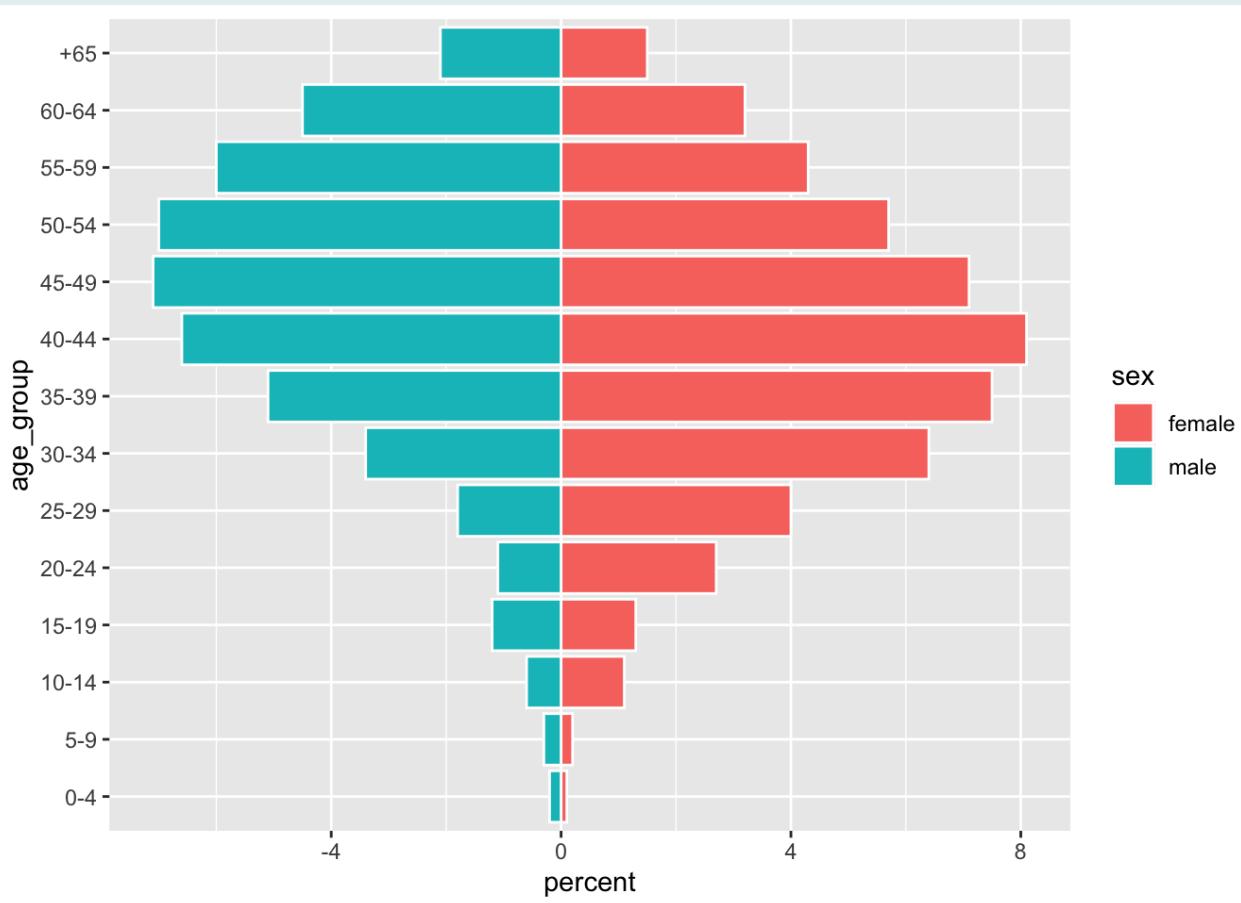
Nous pouvons également créer la même pyramide de population en utilisant le pourcentage total comme valeur de notre axe des y.

```

demo_pyramid_percent <-
# Commencer ggplot
ggplot() +
# Créer un graphique à barres avec geom_bar
geom_col(data = pyramid_data,
          mapping = aes(
            x = age_group,
            y = percent,
            fill = sex),
            colour = "white") +
# Inverser les axes x et y
coord_flip()

demo_pyramid_percent

```



SIDE NOTE



Si votre jeu de données contient le comptage/le pourcentage pour chaque groupe, comme le jeu de données `pyramid_data`, vous pouvez également utiliser `geom_bar`. Cependant, vous devrez passer `stat = "identity"` à l'intérieur de `geom_bar` comme suit :

```

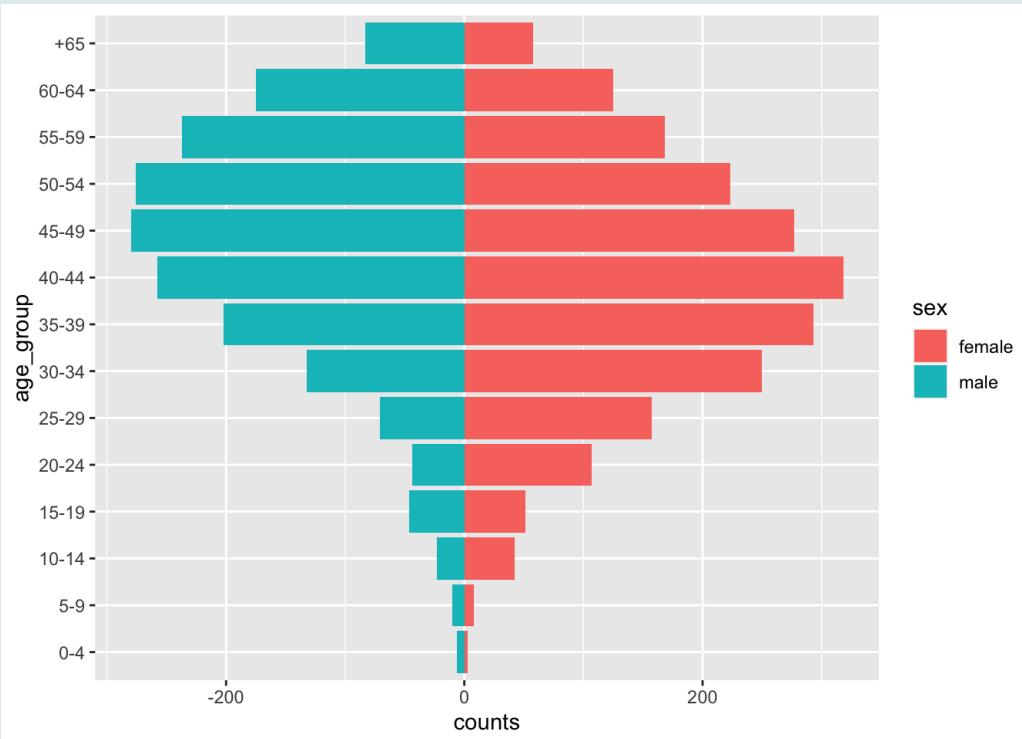
# Commencer ggplot
ggplot(data = pyramid_data,
       mapping = aes(
         x = age_group,
         y = counts,      # Spécifier l'axe des y
         fill = sex),
       colour = "white") +

# Inclure stat = "identity" dans geom_bar
geom_bar(stat = "identity") +

# Inverser les axes x et y
coord_flip()

```

SIDE NOTE



Testons votre compréhension avec les questions à choix multiples suivantes (la clé de réponse se trouve à la fin) :

2. Lors de l'utilisation de `geom_col`, quel type de variable x votre ensemble de données doit-il inclure ?

- a. Variables continues
- b. Variables catégorielles
- c. Variables binaires
- d. Variables ordinaires

3. Quelle fonction `ggplot2` pouvez-vous utiliser pour inverser les axes x et y ?

- a. `coord_flip()`
- b. `x_y_flip()`
- c. `geom_histogram_flip()`

- d. Toutes les options ci-dessus
4. Dans quelle fonction `ggplot2` devriez-vous passer `stat = "identity"` lors de l'utilisation de données déjà agrégées et essayant de créer un diagramme à barres ?
- a. `geom_col`
 - b. `geom_histogram`
 - c. `geom_bar`
 - d. Toutes les options ci-dessus

Testons maintenant votre compréhension avec la question de pratique de codage suivante (la clé de réponse se trouve à la fin) :

Nous utiliserons un ensemble de données nettoyé et préparé contenant la population totale du Zimbabwe en 2016, regroupée par tranche d'âge et sexe.

Commencez par charger l'ensemble de données comme suit :

```
zw_2016 <- readRDS(here::here("data/clean/population_zw_2016.rds"))
```

```
zw_2016
```

```
## # A tibble: 28 × 3
## # Groups:   age_group [14]
##       age_group sex     total_count
##       <fct>      <fct>      <int>
## 1 0-4        Female    1091129
## 2 0-4        Male     -1114324
## 3 10-14      Female    815362
## 4 10-14      Male     -803657
## 5 15-19      Female    803754
## 6 15-19      Male     -792474
## 7 20-24      Female    690940
## 8 20-24      Male     -632342
## 9 25-29      Female    638192
## 10 25-29     Male     -535444
## # ... i 18 more rows
```

Notez que le total masculin est déjà **négatif** !

5. Créez une pyramide démographique pour la population totale du Zimbabwe en 2016 en utilisant la fonction `geom_col` du package `ggplot2`. Assurez-vous d'ajouter une bordure blanche autour de chaque barre !

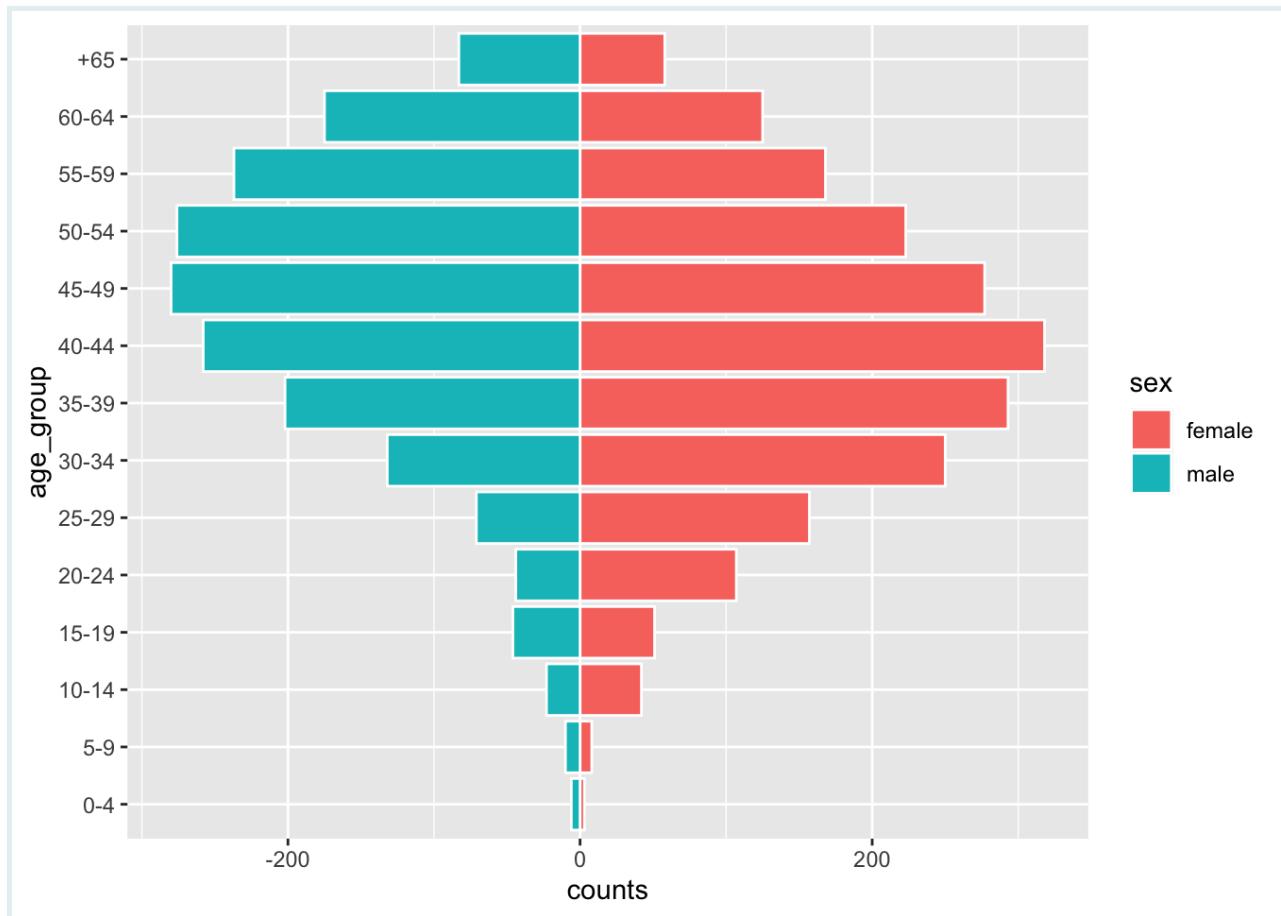
```
Q4_pyramid_zw_2016 <-
# Commencez par ggplot
ggplot() +
# Créez un graphique à barres avec geom_bar
geom_col(data = ____,
aes(x = ____,
y = ____,
fill = ____),
color = ____) +
# Inversez les axes x et y
_____
```

Sure, here is the translated version in French. Remember, certain technical terms and code have been kept in English to ensure they are understood universally by programmers:

Personnalisation de graphique

Jusqu'à présent, vous avez appris à créer une pyramide démographique en utilisant `ggplot2` comme indiqué ci-dessous :

```
demo_pyramid
```



Cependant, pour créer un graphique informatif, un certain niveau de personnalisation du graphique est nécessaire. Par exemple, il est important d'inclure des étiquettes informatives et de redimensionner l'axe des x et des y pour une meilleure visualisation.

Apprenons quelques personnalisations utiles de `ggplot2` !

Étiquettes

Utilisons la pyramide de population que nous avons précédemment créée en utilisant la fonction `geom_col()` et construisons à partir de cela.

Nous pouvons commencer par ajouter un titre informatif, des axes, et une légende à notre graphique :

```

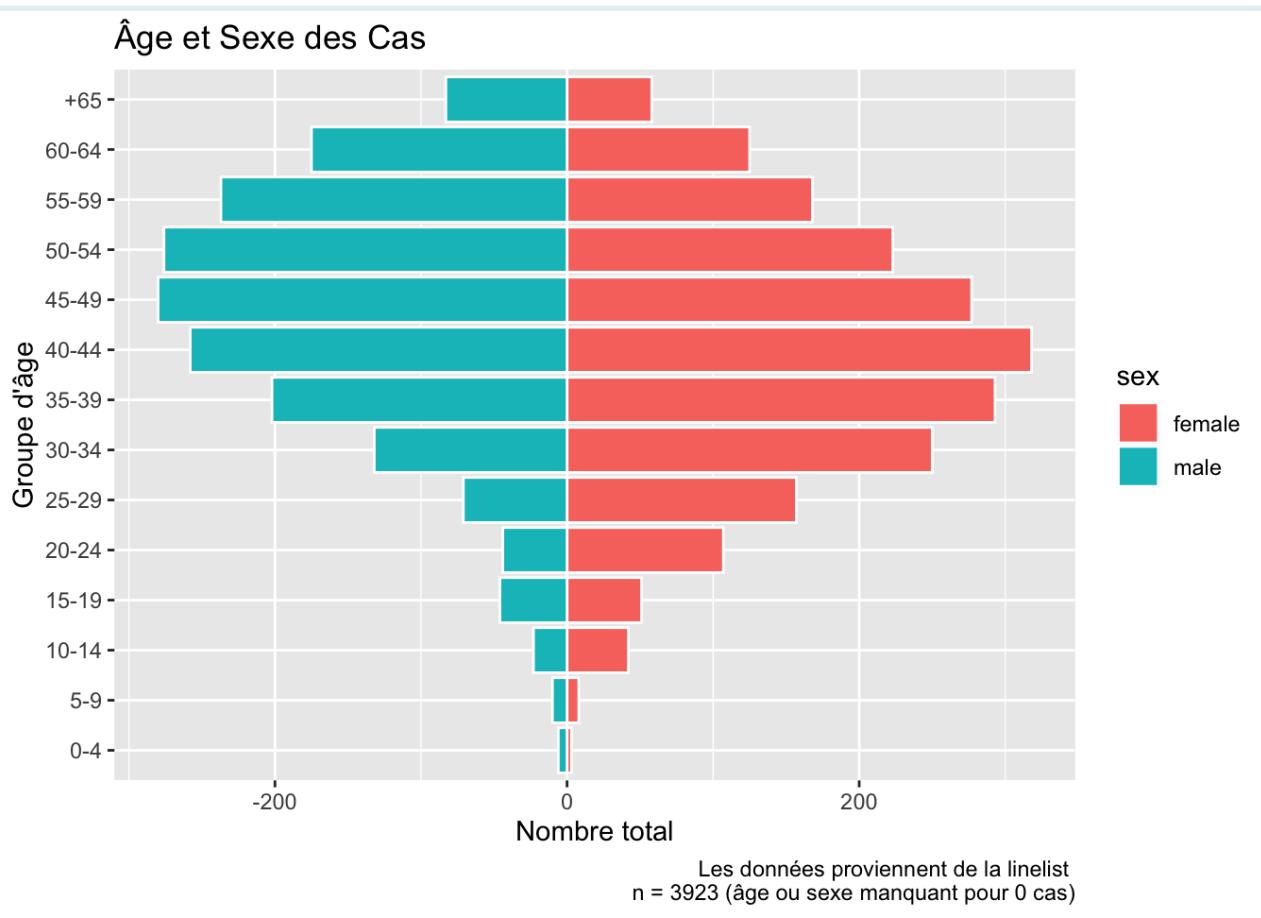
adjusted_labels <-

# Utilisez la pyramide démographique précédente
demo_pyramid + 

# Ajustez les étiquettes
labs(
  title = "Âge et Sexe des Cas",
  x = "Groupe d'âge",
  y = "Nombre total",
  caption = stringr::str_glue("Les données proviennent de la linelist \nn =
  {nrow(hiv_prevalence)} (âge ou sexe manquant pour
  {sum(is.na(hiv_prevalence$sex) | is.na(hiv_prevalence$age_years))} cas)"))
)

adjusted_labels

```



Axe

Redimensionnons nos axes pour nous assurer que les données sont correctement visualisées et comprises.

Nous commencerons par redimensionner l'axe du *nombre total*, ou dans le cas de notre graphique, l'**axe des y**. Pour cela, nous commencerons par identifier les valeurs maximales et minimales et les sauvegarder en tant qu'objets.

```
max_count <- max(pyramid_data$counts, na.rm = T)
min_count <- min(pyramid_data$counts, na.rm = T)

max_count
```

```
## [1] 318
```

```
min_count
```

```
## [1] -280
```

Maintenant que nous avons identifié que la valeur minimale du *nombre total* est **-280** et que la valeur maximale est **318**, nous pouvons l'utiliser pour redimensionner notre axe des y en conséquence.

Dans ce cas particulier, nous voulons redimensionner notre axe des y pour qu'il soit symétrique. Nous prendrons donc la plus grande valeur absolue et l'utiliserons comme limite pour les côtés *positif* et *négatif*.

Dans ce cas, nous utiliserons notre valeur maximale.

```
adjusted_axes <-
  # Utilisez le graphique précédent
  adjusted_labels +
  # Ajustez l'axe des y (nombre total)
  scale_y_continuous(
    # Spécifiez la limite de l'axe des y en utilisant la valeur max et en
    # rendant positif et négatif
    limits = c(max_count * c(-1,1)),
    # Spécifiez comment diviser l'axe des y (basé sur la limite max)
    breaks = seq(-400, 400, 400/5),
    # Rendez les étiquettes de l'axe absolues afin que les étiquettes
    # masculines apparaissent positives
    labels = abs)
```

adjusted_axes

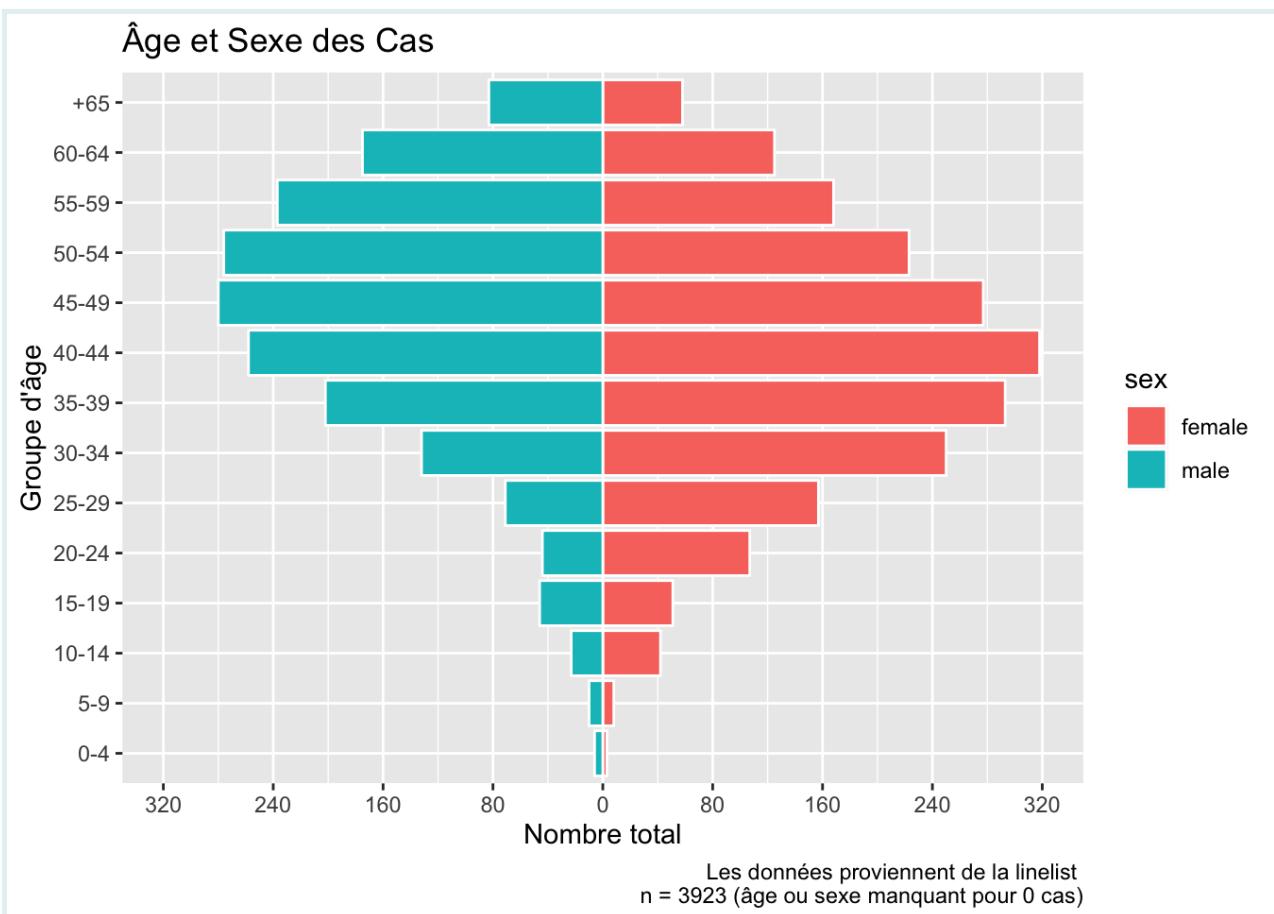


Schéma de Couleurs et Thèmes

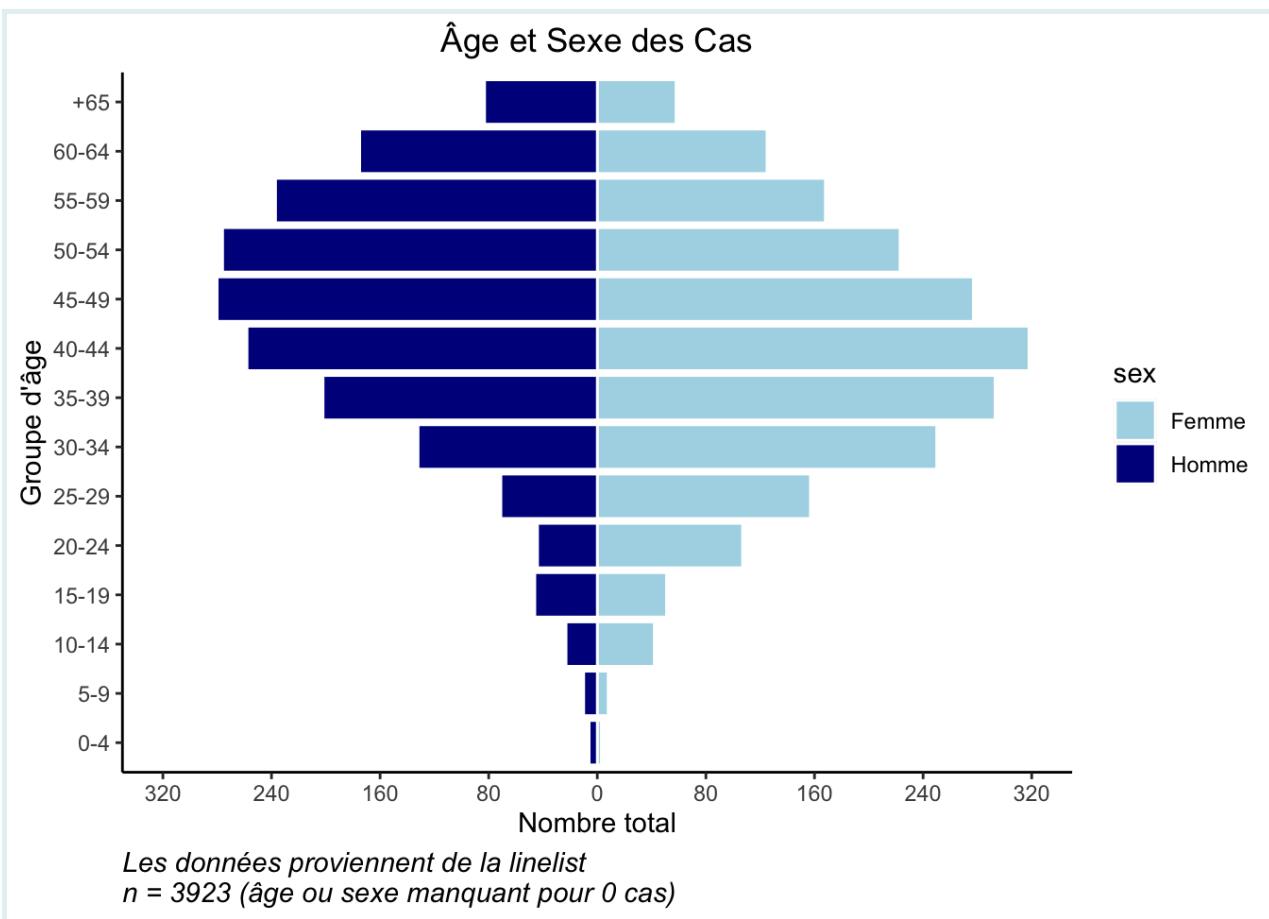
Nous pouvons également effectuer les ajustements nécessaires au schéma de couleurs et au thème du graphique.

Voici un exemple de certaines modifications que nous pouvons effectuer :

```

adjusted_color_theme <-
# Utiliser le graphique précédent
adjusted_axes +
# Désigner manuellement les couleurs et les étiquettes de la légende
scale_fill_manual(
  # Sélectionner la couleur de remplissage selon le sexe
  values = c("female" = "lightblue",
             "male" = "darkblue"),
  # Renommer les étiquettes de la légende
  labels = c("Femme",
             "Homme")) +
# Ajuster les paramètres du thème
theme(
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.background = element_blank(),
  axis.line = element_line(colour = "black"), # rendre la ligne d'axe noire
  plot.title = element_text(hjust = 0.5),      # définir la hauteur du titre
  plot.caption = element_text(hjust = 0, size = 11, face = "italic")) # formater la légende
adjusted_color_theme

```



Récap !

Comme vous pouvez le voir, les pyramides démographiques sont un outil de visualisation essentiel pour comprendre la distribution de maladies spécifiques selon les groupes d'âge et le sexe.

Les concepts appris dans cette leçon peuvent également être appliqués pour créer d'autres types de graphiques nécessitant des sorties à la fois négatives et positives, comme le pourcentage de variation des taux de notification des cas et plus encore.

Maintenant que vous avez compris le concept derrière la création des pyramides démographiques, les possibilités sont infinies ! De la représentation des **cas** par groupe d'âge et sexe sur la **population de référence/vraie** à la représentation de la variation (positive et négative) des interventions dans une population, vous devriez être capable d'appliquer ces concepts pour créer des graphiques épidémiologiques informatifs.

Félicitations pour avoir terminé cette leçon. Nous espérons que vous pourrez maintenant appliquer les connaissances acquises lors de la leçon d'aujourd'hui lors de l'analyse et de la création de rapports d'examen épidémiologique.

Corrigé

1. d
2. b
3. a
4. c
- 5.

```
Q4_pyramid_zw_2016 <-
  ggplot() +
  geom_col(data = zw_2016,
            aes(x = age_group,
                 y = total_count,
                 fill = sex),
            color = "white") +
  coord_flip()
```

Contributeurs

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



SABINA RODRIGUEZ VELÁSQUEZ

Project Manager and Scientific Collaborator, The GRAPH Network
Infectiously enthusiastic about microbes and Global Health



JOY VAZ

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

Références

1. Contenu de leçon ajusté à partir de : Batra, Neale, et al. The Epidemiologist R Handbook. 2021. <https://doi.org/10.5281/zenodo.4752646>
2. Contenu de leçon ajusté à partir de : OMS. Comprendre et utiliser les données sur la tuberculose. 2014. https://apps.who.int/iris/bitstream/handle/10665/129942/9789241548786_eng.pdf
3. Package référencé à partir de : <https://r4epis.netlify.app/>

Présentation des données de santé avec les tableaux gt dans R: Fondamentaux

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
Packages
Présentation du jeu de données
Création de tables simples avec {gt}
Personnalisation des tableaux {gt}
En-tête et pied de tableau
Stub
Colonnes Spanner & sous-colonnes
Renommage des colonnes du tableau
Lignes de résumé
Conclusion
Corrigé

Introduction

Les tables sont un outil puissant pour visualiser les données de manière claire et concise. Avec R et le package `gt`, nous pouvons exploiter l'attrait visuel des tables pour communiquer efficacement des informations clés. Dans cette leçon, nous apprendrons comment construire des tables esthétiquement agréables et personnalisables qui soutiennent les objectifs d'analyse de données.

Objectifs d'apprentissage

- Utiliser la fonction `gt()` pour créer des tables basiques
- Regrouper les colonnes sous des en-têtes spanner
- Renommer les noms des colonnes
- Ajouter des lignes récapitulatives pour les groupes

À la fin, vous serez capable de générer des tables polies et reproductibles comme celle-ci :

Sum of HIV cases in Malawi

from Q1 2019 to Q2 2019

period	New cases		Previous cases	
	Positive	Negative	Positive	Negative
Central Region				
2019 Q1	2004	123018	3682	2562
2019 Q2	1913	116443	3603	1839
2019 Q3	1916	127799	4002	2645
2019 Q4	1691	124728	3754	1052
sum	—	7524.00	491988.00	15041.00
mean	—	1881.00	122997.00	3760.25
				8098.00
				2024.50

Northern Region

2019 Q1	664	36196	1197	675
2019 Q2	582	35315	1084	590
2019 Q3	570	36850	1191	542
2019 Q4	519	34322	1132	346
sum	—	2335.00	142683.00	4604.00
mean	—	583.75	35670.75	1151.00
				2153.00
				538.25

Southern Region

2019 Q1	3531	125480	9937	3358
2019 Q2	3637	130491	10414	3176

Tableau récapitulatif exemple

Packages

Nous utiliserons ces packages :

- `{gt}` pour créer des tables
 - `{tidyverse}` pour la manipulation des données
 - `{here}` pour les chemins de fichiers

```
# Charger les packages  
pacman:::p load(tidyverse, gt, here)
```

Présentation du jeu de données

Nos données proviennent du ** Programme VIH du Malawi ** et couvrent les soins prénataux et le traitement du VIH en 2019. Nous nous concentrerons sur les agrégats régionaux et au niveau des installations par trimestre (disponibles [ici](#)).

```
# Importer les données
hiv_malawi <- read_csv(here::here("data/clean/hiv_malawi.csv"))
```

Explorons les variables :

```
# Les 6 premières lignes  
head(hiv_malawi)
```

```
# Les noms et les types de variables  
glimpse(hiv_malawi)
```

```
## Rows: 17,235
## Columns: 29
## $ region <chr> "Northern R...
## $ zone <chr> "Northern Z...
## $ district <chr> "Chitipa", ...
## $ traditional_authority <chr> "Senior TA ...
## $ facility_name <chr> "Kapenda He...
## $ datim_code <chr> "K9u9BIAaJJ...
## $ system <chr> "e-masterca...
## $ hsector <chr> "Public", ...
## $ period <chr> "2019 Q1", ...
## $ reporting_period <chr> "1st month ...
## $ sub_groups <chr> "All patien...
## $ new_women_registered <dbl> 45, NA, 40, ...
## $ total_women_in_booking_cohort <dbl> NA, 55, NA, ...
```

```

## $ not_tested_for_syphilis <dbl> NA, 45, NA, ...
## $ syphilis_negative <dbl> NA, 10, NA, ...
## $ syphilis_positive <dbl> NA, 0, NA, ...
## $ hiv_status_not_ascertained <dbl> 4, 7, 9, 4, ...
## $ previous_negative <dbl> 0, 0, 0, 0, ...
## $ previous_positive <dbl> 0, 0, 0, 1, ...
## $ new_negative <dbl> 40, 47, 30, ...
## $ new_positive <dbl> 1, 1, 1, 1, ...
## $ not_on_cpt <dbl> NA, 0, NA, ...
## $ on_cpt <dbl> NA, 1, NA, ...
## $ no_ar_vs <dbl> 0, 0, 0, 0, ...
## $ already_on_art_when_starting_anc <dbl> 0, 1, 0, 1, ...
## $ started_art_at_0_27_weeks_of_pregnancy <dbl> 1, 0, 1, 1, ...
## $ started_art_at_28_weeks_of_preg <dbl> 0, 0, 0, 0, ...
## $ no_ar_vs_dispensed_for_infant <dbl> NA, 0, NA, ...
## $ ar_vs_dispensed_for_infant <dbl> NA, 1, NA, ...

```

Les données couvrent les régions géographiques, les établissements de santé, les périodes de temps, les données démographiques des patients, les résultats des tests, les thérapies préventives, les médicaments antirétroviraux, et plus encore. Plus d'informations sur le jeu de données sont dans la section des annexes.

Les variables clés que nous examinerons sont :

1. previous_negative: Le nombre de patients qui ont visité l'établissement de santé au cours de ce trimestre et qui avaient auparavant des tests VIH négatifs.
2. previous_positive: Le nombre de patients (comme ci-dessus) avec des tests VIH positifs précédents.
3. new_negative: Le nombre de patients testant nouvellement négatif pour le VIH.
4. new_positive: Le nombre de patients testant nouvellement positif pour le VIH.

Dans cette leçon, nous allons agréger les données par trimestre et résumer les changements dans les résultats des tests VIH.

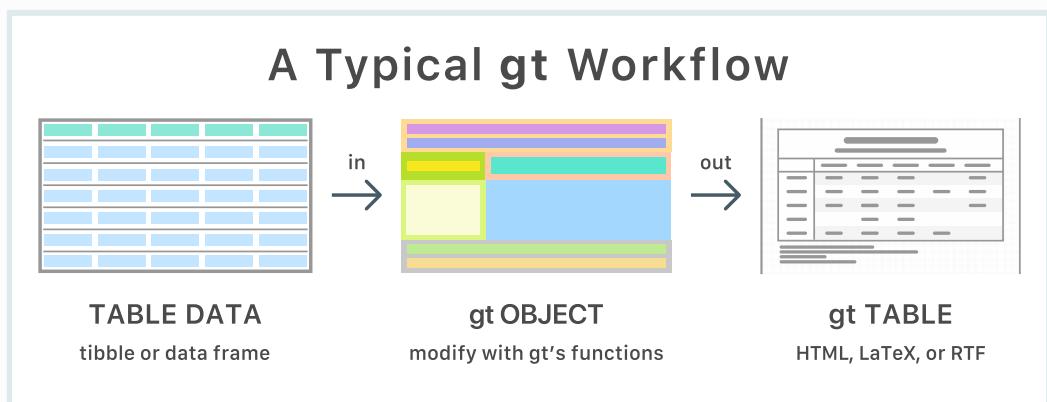
Création de tables simples avec {gt}

La flexibilité, l'efficacité et la puissance de {gt} en font un package redoutable pour la création de tables dans R. Nous explorerons certaines de ses principales caractéristiques dans cette leçon.

KEY POINT



KEY POINT



Le package `{gt}` contient un ensemble de fonctions qui prennent des données brutes en entrée et produisent une table joliment formatée pour une analyse et un rapport ultérieurs.

Pour utiliser efficacement le package `{gt}`, nous devons d'abord transformer nos données dans un format résumé approprié.

Dans le bloc de code ci-dessous, nous utilisons les fonctions de `{dplyr}` pour résumer les tests de VIH dans certains centres de dépistage du Malawi par trimestre. Nous regroupons d'abord les données par période, puis nous additionnons les cas sur plusieurs variables en utilisant `across()` :

```
# Variables à résumer
cols <- c("new_positive", "previous_positive", "new_negative",
         "previous_negative")

# Créer un résumé par trimestre
hiv_malawi_summary <- hiv_malawi %>%
  group_by(period) %>%
  summarize(
    across(all_of(cols), sum) # Résumer toutes les colonnes
  )

hiv_malawi_summary
```

```
## # A tibble: 4 × 5
##   period new_positive previous_positive new_negative
##   <chr>      <dbl>           <dbl>        <dbl>
## 1 2019 Q1     6199            14816       284694
## 2 2019 Q2     6132            15101       282249
## 3 2019 Q3     5907            15799       300529
## 4 2019 Q4     5646            15700       291622
## # i 1 more variable: previous_negative <dbl>
```

Ceci agrège les données de manière appropriée pour les passer à `{gt}` afin de générer un tableau résumé propre.

Pour créer une table simple à partir des données agrégées, nous pouvons ensuite appeler la fonction `gt()` :

```
hiv_malawi_summary %>%  
  gt()
```

period	new_positive	previous_positive	new_negative	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

Comme vous pouvez le voir, le formatage de table par défaut est assez simple et non raffiné. Cependant, `{gt}` offre de nombreuses options pour personnaliser et embellir la sortie de table. Nous approfondirons ces aspects dans la prochaine section.

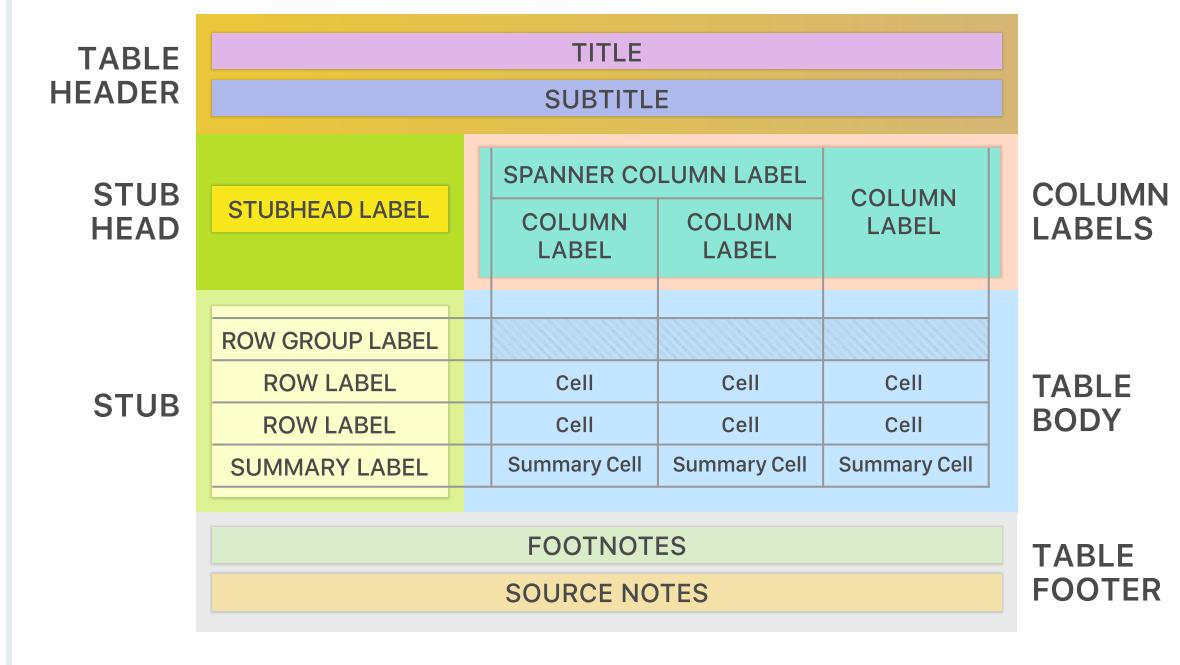
Voici la traduction en français du fragment de tutoriel Rmd, en gardant la syntaxe Rmd et le code valide :

Personnalisation des tableaux `{gt}`

Le package `{gt}` permet une personnalisation complète des tableaux grâce à son cadre de « grammaire des tableaux ». C'est similaire à la façon dont la grammaire graphique de `{ggplot2}` fonctionne pour les graphiques.

Pour tirer pleinement parti de `{gt}`, il est utile de comprendre certains éléments clés de sa grammaire.

The Parts of a gt Table



Comme on le voit dans la figure du site du package, les principaux composants d'un tableau {gt} sont :

-En-tête du tableau: Contient un titre et un sous-titre facultatifs

-Colonne d'étiquette: Étiquettes de lignes qui identifient chaque ligne

-En-tête de colonne d'étiquette: Regroupement et étiquettes facultatifs pour les lignes de colonne d'étiquette

-Étiquettes de colonne: En-têtes pour chaque colonne

-Corps du tableau: Les principales cellules de données du tableau

-Pied de tableau: Notes de bas de page et notes de source facultatives

Comprendre cette anatomie nous permet de construire systématiquement des tableaux {gt} en utilisant sa grammaire.

En-tête et pied de tableau

Le tableau de base que nous avions peut maintenant être mis à jour avec plus de composants.

Les tableaux deviennent plus informatifs et professionnels avec l'ajout d'en-têtes, de notes de source et de notes de bas de page. Nous pouvons facilement améliorer le tableau de base d'avant en ajoutant ces éléments à l'aide des fonctions {gt}.

Pour créer un en-tête, nous utilisons `tab_header()` et spécifions un `title` et `subtitle`. Cela donne au lecteur le contexte de ce que montre le tableau.

```
hiv_malawi_summary %>%
  gt() %>%
  tab_header(
    title = "Dépistage du VIH au Malawi",
    subtitle = "T1 à T4 2019"
  )
```

Dépistage du VIH au Malawi

T1 à T4 2019

period	new_positive	previous_positive	new_negative	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

Nous pouvons ajouter un pied de page avec la fonction `tab_source_note()` pour citer la provenance des données :

```
hiv_malawi_summary %>%
  gt() %>%
  tab_header(
    title = "Dépistage du VIH au Malawi",
    subtitle = "T1 à T4 2019"
  ) %>%
  tab_source_note("Source : Programme VIH du Malawi")
```

Dépistage du VIH au Malawi

T1 à T4 2019

period	new_positive	previous_positive	new_negative	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

Source : Programme VIH du Malawi

Les notes de bas de page sont utiles pour fournir des détails supplémentaires sur certains points de données ou étiquettes. La fonction `tab_footnote()` attache les notes de bas de page aux cellules de tableau indiquées. Par exemple, nous pouvons annoter les colonnes de diagnostic :

```
hiv_malawi_summary %>%
  gt() %>%
  tab_header(
    title = "Dépistage du VIH au Malawi",
    subtitle = "T1 à T2 2019"
  ) %>%
  tab_source_note("Source : Programme VIH du Malawi") %>%
  tab_footnote(
    footnote = "Nouveau diagnostic",
    locations = cells_column_labels(columns = c(new_positive, new_negative))
  )
```

Dépistage du VIH au Malawi

T1 à T2 2019

period	new_positive ¹	previous_positive	new_negative ¹	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

¹ Nouveau diagnostic

Source : Programme VIH du Malawi

Ces petits ajouts améliorent grandement l'apparence professionnelle et informative des tableaux.

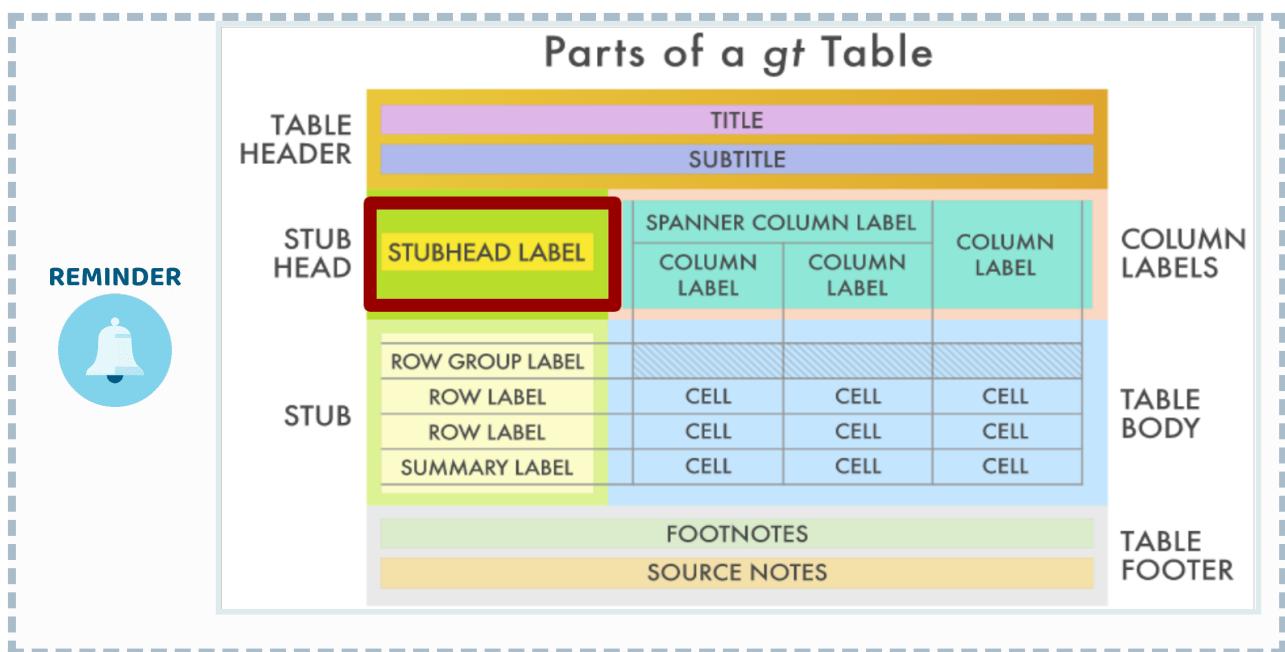
Stub

Le stub est la section gauche d'un tableau contenant les étiquettes de ligne. Elles fournissent un contexte pour les données de chaque ligne.

REMINDER



Cette image montre le composant stub d'un tableau `{gt}`, marqué par un carré rouge.



Dans notre tableau de cas de VIH, la colonne `period` contient les étiquettes de ligne que nous voulons utiliser. Pour générer un stub, nous spécifions cette colonne dans `gt()` en utilisant l'argument `rowname_col`:

```
hiv_malawi_summary %>%
  gt(rowname_col = "period") %>%
  tab_header(
    title = "Dépistage du VIH au Malawi",
    subtitle = "T1 à T2 2019"
  ) %>%
  tab_source_note("Source: Programme VIH du Malawi")
```

Dépistage du VIH au Malawi

T1 à T2 2019

	new_positive	previous_positive	new_negative	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

Source: Programme VIH du Malawi

Notez que le nom de la colonne passé à `rowname_col` doit être entre guillemets.

Pour plus de commodité, sauvegardons le tableau dans une variable `t1`:

```
t1 <- hiv_malawi_summary %>%
  gt(rownames_col = "period") %>%
  tab_header(
    title = "Dépistage du VIH au Malawi",
    subtitle = "T1 à T2 2019"
  ) %>%
  tab_source_note("Source: Programme VIH du Malawi")

t1
```

Dépistage du VIH au Malawi
T1 à T2 2019

	new_positive	previous_positive	new_negative	previous_negative
2019 Q1	6199	14816	284694	6595
2019 Q2	6132	15101	282249	5605
2019 Q3	5907	15799	300529	6491
2019 Q4	5646	15700	291622	6293

Source: Programme VIH du Malawi

Colonnes Spanner & sous-colonnes

Pour mieux structurer notre tableau, nous pouvons regrouper des colonnes liées sous des “spanners”. Les spanners sont des titres qui couvrent plusieurs colonnes, fournissant une organisation catégorielle de niveau supérieur. Nous pouvons le faire avec la fonction `tab_spinner()`.

Créons deux colonnes spanner pour les tests nouveaux et précédents. Nous commencerons par le spanner “Nouveaux tests” afin que vous puissiez observer la syntaxe :

```
t1 %>%
  tab_spinner(
    label = "Nouveaux tests",
    columns = starts_with("new") # sélectionne les colonnes commençant par
      "new"
  )
```

Dépistage du VIH au Malawi

T1 à T2 2019

	Nouveaux tests			
	new_positive	new_negative	previous_positive	previous_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: Programme VIH du Malawi

L'argument `columns` nous permet de sélectionner les colonnes pertinentes, et l'argument `label` prend en entrée l'étiquette du spanner.

Ajoutons maintenant les deux spanners :

```
# Enregistre le tableau dans t2 pour un accès facile
t2 <- t1 %>%
  # Premier spanner pour "Nouveaux tests"
  tab_spinner(
    label = "Nouveaux tests",
    columns = starts_with("new")
  ) %>%
  # Second spanner pour "Tests précédents"
  tab_spinner(
    label = "Tests précédents",
    columns = starts_with("prev")
  )
t2
```

Dépistage du VIH au Malawi

T1 à T2 2019

	Nouveaux tests		Tests précédents	
	new_positive	new_negative	previous_positive	previous_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: Programme VIH du Malawi

Notez que la fonction `tab_spanner` a automatiquement réarrangé les colonnes de manière appropriée.

Question 1 : Le but des Spanners

Quel est le but de l'utilisation de "colonnes spanner" dans un tableau `gt` ?

- A. Appliquer des styles CSS personnalisés à des colonnes spécifiques.
- B. Créer des colonnes de groupe et augmenter la lisibilité.
- C. Formater la taille de la police de toutes les colonnes de manière uniforme.
- D. Trier les données par ordre croissant.



Question 2 : Crédit de Spanners

En utilisant le cadre de données `hiv_malawi`, créez un tableau `gt` qui affiche un résumé de la `somme` des cas "nouveaux_positifs" et "précédents_positifs" pour chaque région. Créez des en-têtes spanner pour étiqueter ces deux colonnes de résumé. Pour y parvenir, remplissez les parties manquantes du code ci-dessous :



```
region_summary <- hiv_malawi %>%
  group_by(region) %>%
  summarize(
    _____ (
      c(nouveaux_positifs, précédents_positifs),
      _____
    )
  )

# Créer un tableau gt avec des en-têtes spanner
summary_table_spanners <- region_summary %>%
  _____ %>%
  _____ (
    label = "Cas positifs",
    _____ = c(nouveaux_positifs, précédents_positifs)
  )
```

Renommage des colonnes du tableau

Les noms des colonnes contiennent actuellement des préfixes inutiles comme “new_” et “previous_”. Pour une meilleure lisibilité, nous pouvons les renommer en utilisant `cols_label()`.

`cols_label()` prend un ensemble d’anciens noms à apparter (du côté gauche d’une tilde, ~) et de nouveaux noms pour les remplacer (du côté droit de la tilde). Nous pouvons utiliser `contains()` pour sélectionner des colonnes avec “positive” ou “negative” :

```
t3 <- t2 %>%
  cols_label(
    contains("positive") ~ "Positive",
    contains("negative") ~ "Negative"
  )

t3
```

Dépistage du VIH au Malawi

T1 à T2 2019

	Nouveaux tests		Tests précédents	
	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: Programme VIH du Malawi

Ceci renomme les colonnes d'une manière plus propre.

`cols_label()` accepte plusieurs aides à la sélection de colonnes comme `contains()`, `starts_with()`, `ends_with()` etc. Celles-ci proviennent du paquet `tidyselect` et offrent une flexibilité dans le renommage.



`cols_label()` a d'autres fonctions d'identification comme `contains()` qui fonctionnent de manière similaire et sont identiques aux aides de `tidyselect`, celles-ci incluent également :

- `starts_with()`: Commence par un préfixe exact.
- `ends_with()`: Se termine par un suffixe exact.
- `contains()`: Contient une chaîne de caractères littérale.
- `matches()`: Correspond à une expression régulière.
- `num_range()`: Correspond à une plage numérique comme x01, x02, x03.

Ces aides sont utiles, en particulier dans le cas de la sélection de plusieurs colonnes.

Pour en savoir plus sur la fonction `cols_label()`, vous pouvez consulter ici : https://rstudio.com/reference/cols_label.html

Question 3 : étiquettes de colonnes

Quelle fonction est utilisée pour changer les étiquettes ou les noms des colonnes dans un tableau `gt` ?

PRACTICE



- A. ``tab_header()``
- B. ``tab_style()``
- C. ``tab_options()``
- D. ``tab_relabel()``

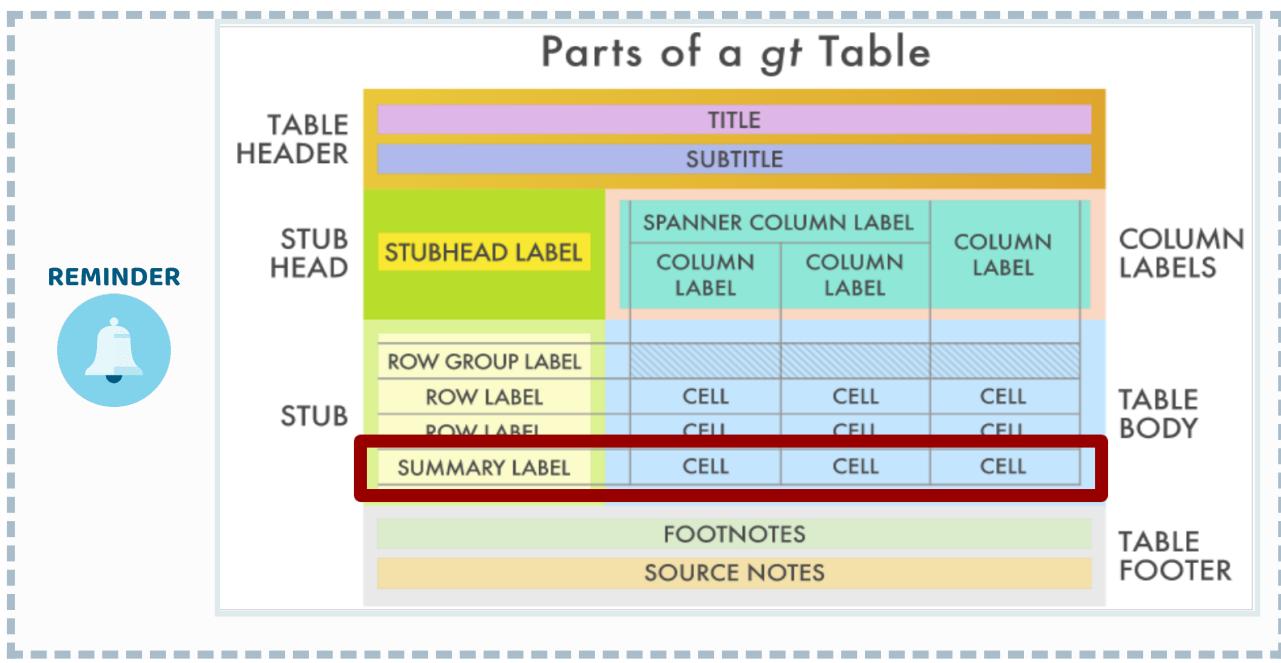
Lignes de résumé

Prenons les mêmes données avec lesquelles nous avons commencé au début de cette leçon et, au lieu de regrouper uniquement par période (trimestres), regroupons à la fois par période et par région. Nous faisons cela pour illustrer la puissance des fonctionnalités de résumé dans `gt` : les tableaux récapitulatifs.

REMINDER



Rappel {gt} - Lignes de résumé Cette image montre le composant des lignes de résumé d'un tableau `{gt}`, clairement indiqué dans un carré rouge. Les lignes de résumé fournissent des données agrégées ou des résumés statistiques des données contenues dans les colonnes correspondantes.



Tout d'abord, recréons les données :

```
summary_data_2 <- hiv_malawi %>%
  group_by(
    # Notez l'ordre des variables par lesquelles nous regroupons.
    region,
    period
  ) %>%
  summarise(
    across(all_of(cols), sum)
  ) %>%
  gt()

summary_data_2
```

period	new_positive	previous_positive	new_negative	previous_negative
Central Region				
2019 Q1	2004	3682	123018	2562
2019 Q2	1913	3603	116443	1839
2019 Q3	1916	4002	127799	2645
2019 Q4	1691	3754	124728	1052
Northern Region				
2019 Q1	664	1197	36196	675
2019 Q2	582	1084	35315	590
2019 Q3	570	1191	36850	542
2019 Q4	519	1132	34322	346
Southern Region				
2019 Q1	3531	9937	125480	3358
2019 Q2	3637	10414	130491	3176
2019 Q3	3421	10606	135880	3304
2019 Q4	3436	10814	132572	4895

WATCH OUT



L'ordre dans la fonction `group_by()` affecte les groupes de lignes dans le tableau `gt`.

Deuxièmement, réincorporons tous les changements que nous avons précédemment effectués dans ce tableau :

```
# sauvegardant les progrès dans l'objet t4

t4 <- summary_data_2 %>%
  tab_header(
    title = "Somme des tests VIH au Malawi",
    subtitle = "du T1 2019 au T4 2019"
  ) %>%
  tab_source_note("Source des données : Programme VIH du Malawi") %>%
    tab_spinner(
      label = "Nouveaux tests",
      columns = starts_with("new") # sélectionne les colonnes commençant par
        "new"
    ) %>%
    # création du premier spinner pour les tests précédents
  tab_spinner(
    label = "Tests précédents",
    columns = starts_with("prev") # sélectionne les colonnes commençant par
      "prev"
  ) %>%
  cols_label(
    # localiser ### assigner
    contains("positive") ~ "Positif",
    contains("negative") ~ "Négatif"
  )
)

t4
```

Somme des tests VIH au Malawi

du T1 2019 au T4 2019

period	Nouveaux tests		Tests précédents	
	Positif	Négatif	Positif	Négatif
Central Region				
2019 Q1	2004	123018	3682	2562
2019 Q2	1913	116443	3603	1839
2019 Q3	1916	127799	4002	2645
2019 Q4	1691	124728	3754	1052
Northern Region				
2019 Q1	664	36196	1197	675
2019 Q2	582	35315	1084	590
2019 Q3	570	36850	1191	542
2019 Q4	519	34322	1132	346
Southern Region				
2019 Q1	3531	125480	9937	3358
2019 Q2	3637	130491	10414	3176
2019 Q3	3421	135880	10606	3304
2019 Q4	3436	132572	10814	4895
Source des données : Programme VIH du Malawi				

Maintenant, que faire si nous voulons visualiser sur le tableau un résumé de chaque variable pour chaque groupe de régions ? Plus précisément, nous voulons voir la somme et la moyenne pour les 4 colonnes que nous avons pour chaque région.

N'oubliez pas que nos 4 colonnes d'intérêt sont : "new_positive", "previous_positive", "new_negative", et "previous_negative". Nous n'avons changé les labels de ces colonnes que dans la table `gt` et non dans le `data.frame` lui-même, nous pouvons donc utiliser les noms de ces colonnes pour indiquer à `gt` où appliquer la fonction de résumé. De plus, nous avons déjà stocké les noms de ces 4 colonnes dans l'objet `cols` donc nous allons l'utiliser à nouveau ici.

Pour atteindre cet objectif, nous allons utiliser la fonction pratique `summary_rows` où nous fournissons explicitement les colonnes que nous voulons résumer, et les fonctions avec lesquelles nous voulons résumer, dans notre cas c'est `sum` et `mean`. Notez que nous assignons le nom de la nouvelle ligne (non citée) à un nom de fonction ("cité").

```
t5 <- t4 %>%
  summary_rows(
    columns = cols, #using columns = 3:6 also works
    fns = list(
      TOTAL = "sum",
      MOYENNE = "mean"
    )
  )
t5
```

Somme des tests VIH au Malawi

du T1 2019 au T4 2019

period	Nouveaux tests		Tests précédents	
	Positif	Négatif	Positif	Négatif
Central Region				
2019 Q1	2004	123018	3682	2562
2019 Q2	1913	116443	3603	1839
2019 Q3	1916	127799	4002	2645
2019 Q4	1691	124728	3754	1052
sum	—	7524.00	491988.00	15041.00
mean	—	1881.00	122997.00	3760.25
Northern Region				
2019 Q1	664	36196	1197	675
2019 Q2	582	35315	1084	590
2019 Q3	570	36850	1191	542
2019 Q4	519	34322	1132	346
sum	—	2335.00	142683.00	4604.00
mean	—	583.75	35670.75	1151.00
Southern Region				
2019 Q1	3531	125480	9937	3358
2019 Q2	3637	130491	10414	3176
2019 Q3	3421	135880	10606	3304
2019 Q4	3436	132572	10814	4895
sum	—	14025.00	524423.00	41771.00
mean	—	3506.25	131105.75	10442.75

Source des données : Programme VIH du Malawi

Question 4 : lignes résumées

Quelle est la bonne réponse (ou les bonnes réponses) si vous deviez résumer l'écart type des lignes des colonnes "new_positive" et "previous_negative" uniquement?

- A. Utilisez `summary_rows()` avec l'argument `columns` défini sur "new_positive" et "previous_negative" et l'argument `fns` défini sur "sd".

```
# Option A
your_data %>%
  summary_rows(
    columns = c("new_positive", "previous_negative"),
    fns = "sd"
  )
```

- B. Utilisez `summary_rows()` avec l'argument `columns` défini sur "new_positive" et "previous_negative" et l'argument `fns` défini sur "summarize(sd)".

```
# Option B
your_data %>%
  summary_rows(
    columns = c("new_positive", "previous_negative"),
    fns = summarize(sd)
  )
```

- C. Utilisez `summary_rows()` avec l'argument `columns` défini sur "new_positive" et "previous_negative" et l'argument `fns` défini sur `list(ECART_TYPE = "sd")`.

```
# Option C
your_data %>%
  summary_rows(
    columns = c("new_positive", "previous_negative"),
    fns = list(ECART_TYPE = "sd")
  )
```

- D. Utilisez `summary_rows()` avec l'argument `columns` défini sur "new_positive" et "previous_negative" et l'argument `fns` défini sur "standard_deviation".

```
# Option D
your_data %>%
  summary_rows(
    columns = c("new_positive", "previous_negative"),
    fns = "standard_deviation"
  )
```

Conclusion

Dans la leçon d'aujourd'hui, nous nous sommes attaqués aux tables de données dans R en utilisant `gt`. Nous avons commencé par définir des objectifs clairs, présenté les packages que nous utiliserons et découvert notre jeu de données. Ensuite, nous avons mis la main à la pâte en créant des tables simples. Nous avons appris à organiser nos données proprement en utilisant des colonnes `spanner` et en ajustant les étiquettes des colonnes pour rendre les choses parfaitement claires et cohérentes. Nous avons ensuite conclu avec quelques résumés de table astucieux. Ce sont les bases de la création de tables dans R et `gt`, et elles seront très utiles à mesure que nous poursuivrons notre voyage pour créer des tables engageantes et informatives dans R.

Corrigé

1. B

2.

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

# résumer les données d'abord
district_summary <- hiv_malawi %>%
  group_by(region) %>%
  summarize(
    across( #1
      c(new_positive, previous_positive),
      sum #2
    )
  )

# Créer une table gt avec des en-têtes spanner
summary_table_spanners <- district_summary %>%
  gt() %>% #3
  tab_spanner( #4
    label = "Cas positifs",
    columns = c(new_positive, previous_positive) #5
  )
```

3. D

4. C

Contributeurs

Les membres de l'équipe suivants 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

Références

1. Tom Mock, "The Definite Cookbook of {gt}" (2021), The Mockup Blog, <https://themockup.blog/static/resources/gt-cookbook.html#introduction>.
2. Tom Mock, "The Grammar of Tables" (May 16, 2020), The Mockup Blog, <https://themockup.blog/posts/2020-05-16-gt-a-grammar-of-tables/#add-titles>.
3. RStudio, "Introduction to Creating gt Tables," Official {gt} Documentation, <https://gt.rstudio.com/articles/intro-creating-gt-tables.html>.
4. Fleming, Jessica A., Alister Munthali, Bagrey Ngwira, John Kadzandira, Monica Jamili-Phiri, Justin R. Ortiz, Philipp Lambach, et al. 2019. "Maternal Immunization in Malawi: A Mixed Methods Study of Community Perceptions, Programmatic Considerations, and Recommendations for Future Planning." *Vaccine* 37 (32): 4568-75. <https://doi.org/10.1016/j.vaccine.2019.06.020>.

This work is licensed under the [Creative Commons Attribution Share Alike](#) license.



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
Objectifs d'apprentissage
Packages
Précédemment dans pt1
Jeu de données
Thèmes
Formatage des valeurs dans le tableau
Formatage conditionnel
Police et texte
Bordures
Corrigé
Ressources et packages externes

Introduction

La leçon précédente 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, 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 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					
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

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

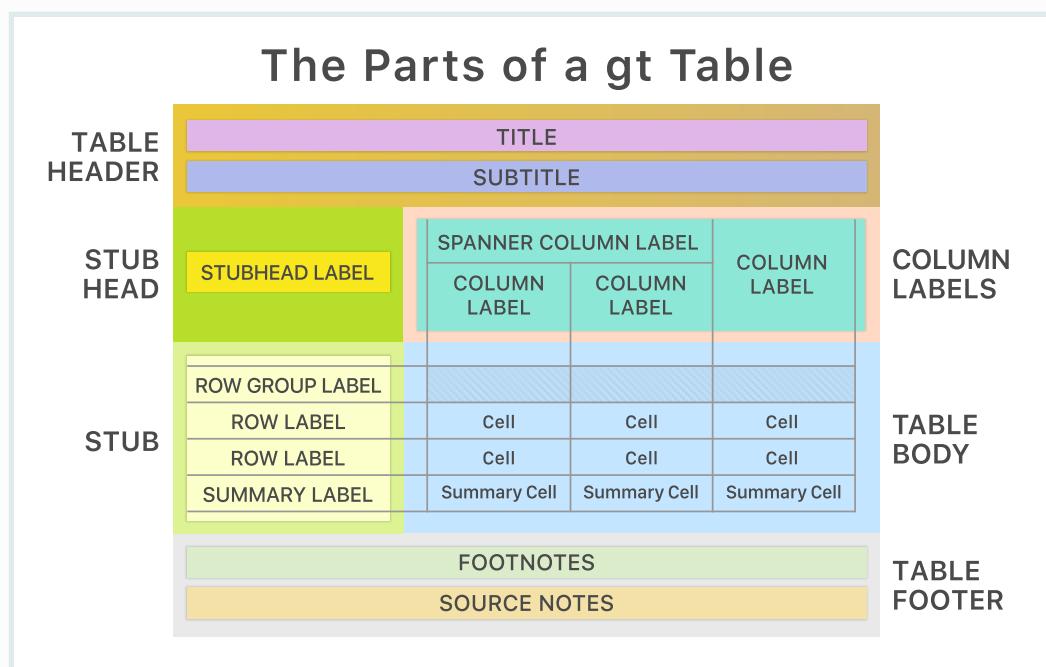
RECAP



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 :



- 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

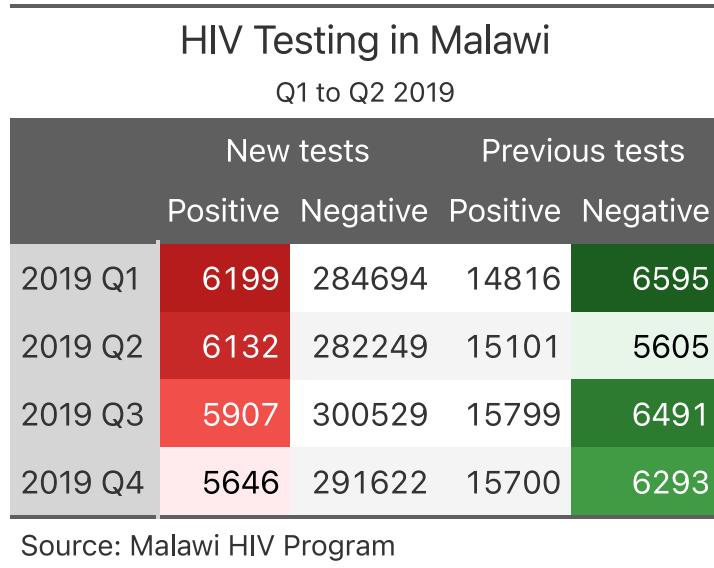
SIDE NOTE



`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"
  )
```



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 :

```
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 référerons 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 styliser.

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
    )
  )
)

```

Police 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.

```
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 :

```

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



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.

PRACTICE



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.

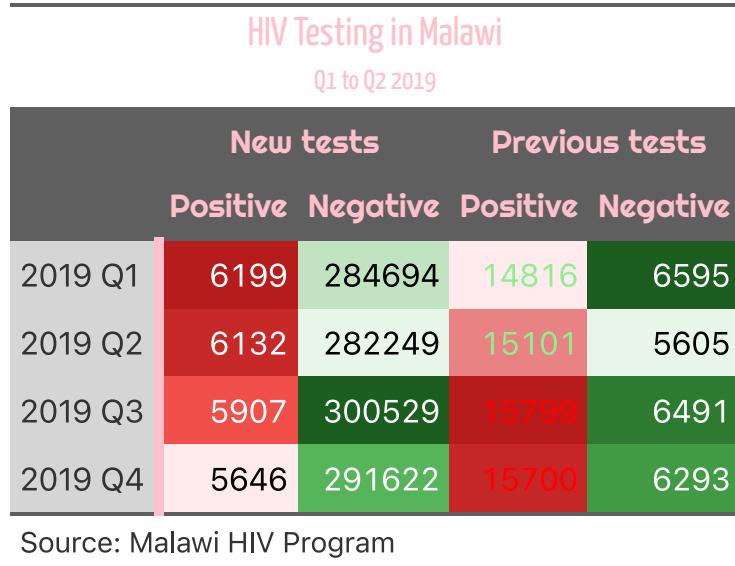
```
data %>%
  gt() %>%
  tab_style(
    style = cells_header(),
    css = "font-size: 16px;"
  )
```

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
```



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

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")
```

CHALLENGE



B.

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

C.

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

D.

CHALLENGE



```
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

Cartographie épidémique avec R : Introduction

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

Création de cartes choroplèthes avec ggplot2
Introduction
Préparation des données
Création d'une belle carte choroplète avec ggplot2
Échelle des couleurs
Facet Wrap vs. Grid
Conclusion
Dernières réflexions
Références
Récapitulatif
Solution

Création de cartes choroplèthes avec ggplot2

Objectifs d'apprentissage

Dans cette leçon, vous apprendrez principalement à utiliser le package `ggplot2`, en particulier les fonctions `ggplot()` et `geom_sf()`, pour créer de belles cartes choroplèthes.

Plus spécifiquement, vous apprendrez comment :

1. Faire correspondre des données brutes à des polygones
 - Obtenir des limites/polynômes
 - Obtenir des données liées aux maladies
 - Joindre par niveau administratif
 2. Appliquer une échelle de couleurs pour les variables continues et discrètes
 - Continues
 - Discrètes
 3. Appliquer Facet Wrap et Grid
 - Créer de petits multiples et utiliser `facet_wrap()`
 - Créer de petits multiples et utiliser `facet_grid()`
-

Introduction

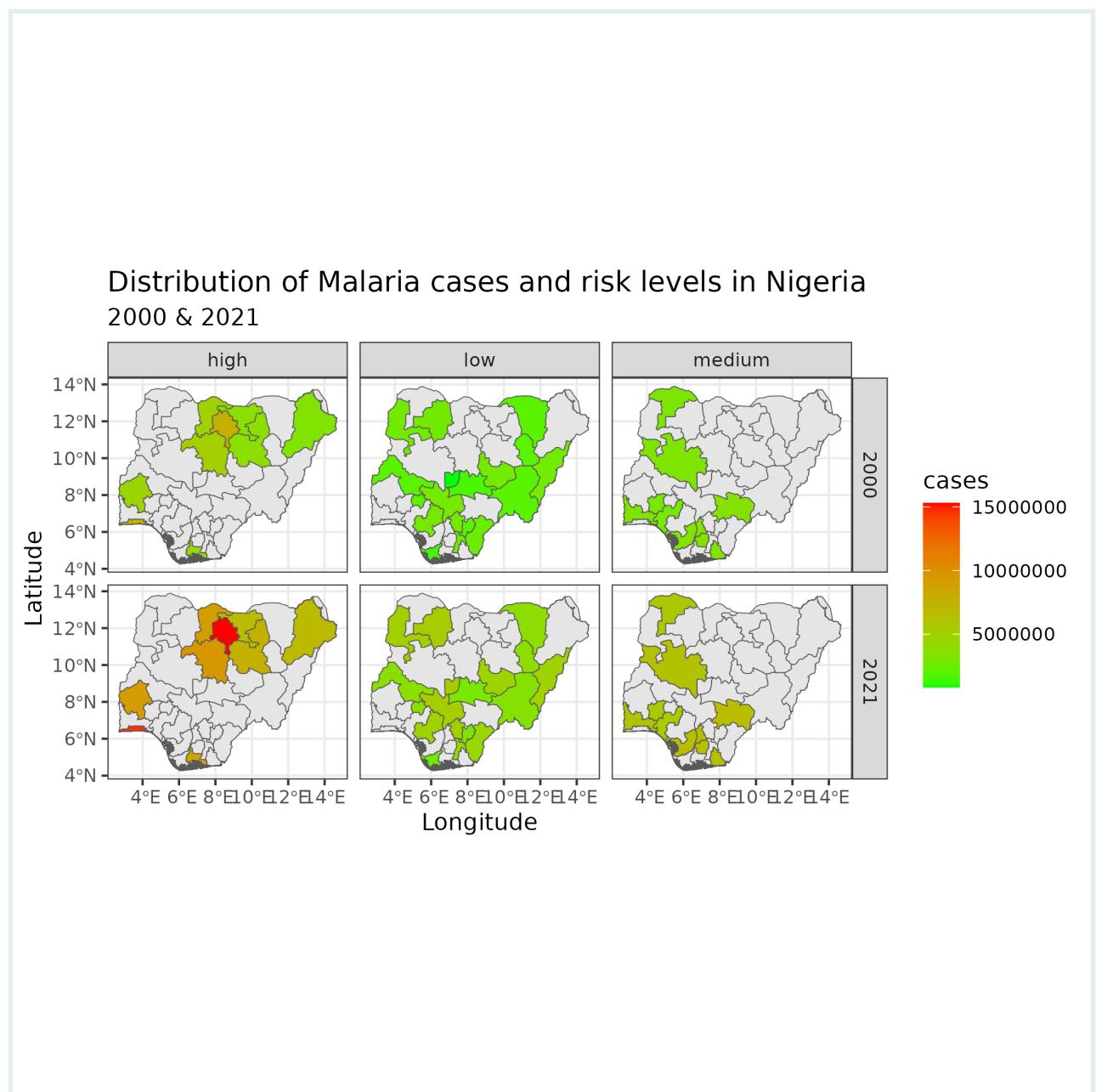
Une carte choroplète est une carte thématique dans laquelle les régions géographiques sont ombrées ou hachurées en proportion de la valeur d'une variable représentée. Cette variable peut être un indicateur épidémiologique tel que la prévalence des maladies ou le taux de mortalité. Les cartes choroplèthes sont particulièrement utiles pour visualiser les schémas spatiaux et les variations à travers différentes régions.

Les composantes essentielles d'une carte choroplète comprennent :

Régions géographiques : Il s'agit des zones qui seront représentées sur la carte, telles que les pays, les États, les districts ou toute autre division géographique.

Valeurs des données : Ce sont les valeurs associées à chaque région géographique qui seront représentées sur la carte, y compris la densité de population, la prévalence, l'incidence, le taux de mortalité, etc.

Échelle de couleurs : Il s'agit de la gamme de couleurs utilisée pour représenter les différentes valeurs des données. Généralement, on utilise un dégradé de couleurs, les couleurs plus claires représentant des valeurs plus faibles et les couleurs plus foncées représentant des valeurs plus élevées.



SIDE NOTE



Les cartes choroplèthes présentent plusieurs avantages notables :

- **Clarté visuelle** : Elles fournissent une représentation visuelle claire de la distribution spatiale des données, ce qui facilite l'identification des schémas et des tendances.
- **Facilité de compréhension** : Elles sont intuitives et faciles à comprendre, même pour les personnes sans connaissance approfondie dans le domaine.
- **Polyvalence** : Elles peuvent être utilisées pour représenter une grande variété de données, ce qui les rend adaptées à diverses applications.

Cependant, il existe également des limitations importantes à prendre en compte :

- **Sensibilité aux données** : L'apparence de la carte peut être considérablement affectée par le choix de la classification des données et de l'échelle de couleurs.
- **Biais de la taille géographique** : Les régions plus grandes peuvent sembler plus prédominantes sur la carte, ce qui entraîne un biais visuel.

Dans la section suivante, vous apprendrez comment créer une carte choroplète à l'aide du package `ggplot2` dans R.

Packages

Cet extrait de code montre le chargement des packages nécessaires pour les analyses. Dans ce guide, nous mettons l'accent sur `p_load()` du package `pacman`, qui installe le package si nécessaire et le charge pour l'utilisation. Vous pouvez également charger des packages installés avec `library()` de base R. Consultez la page sur les bases de R pour plus d'informations sur la gestion des packages en R.

```
# Charger les packages
if(!require(pacman)) install.packages("pacman")
pacman::p_load(malariaAtlas,
                ggpplot2,
                geodata,
                dplyr,
                here,
                readr,
                sf,
                patchwork)

# Désactiver la notation scientifique
options(scipen=10000)
```

Préparation des données

Avant de créer une carte choroplèthe, il est essentiel de préparer les données. Les données doivent contenir les régions géographiques et les valeurs que vous souhaitez visualiser.

Dans cette section, vous passerez par le processus de préparation des données, qui comprend les étapes suivantes :

- 1. Importation des données polygonales** : Il s'agit des données géographiques qui contiennent les limites de chaque région que vous souhaitez inclure dans votre carte.
- 2. Importation des données attributaires** : Il s'agit des données contenant les valeurs que vous souhaitez visualiser sur la carte, telles que la prévalence des maladies, la densité de la population, etc.
- 3. Joindre les données polygonales et attributaires** : Cette étape consiste à fusionner les données polygonales avec les données attributaires en fonction d'un identifiant commun, tel que le niveau administratif ou le nom de la région. Cela créera un seul ensemble de données contenant à la fois les limites géographiques et les valeurs de données correspondantes.

Maintenant, passons en revue chaque étape en détail !

Étape 1 : Importation des données polygonales

Les polygones sont des formes fermées avec trois côtés ou plus. Dans les données spatiales, les polygones sont utilisés pour représenter des zones telles que les limites d'une ville, d'un lac ou d'un type d'utilisation des terres. Ils sont essentiels dans les systèmes d'information géographique (SIG) pour des tâches telles que la cartographie, l'analyse spatiale et la classification de l'occupation des terres.

SIDE NOTE

SIDE NOTE

Les fichiers shapefile sont un format courant pour stocker des données spatiales. Ils se composent d'au moins trois fichiers avec les extensions .shp (forme), .shx (index) et .dbf (données attributaires).

En R, vous pouvez charger des fichiers shapefile à l'aide du package `sf`. Dans notre leçon d'aujourd'hui, nous travaillerons avec des données sur le paludisme provenant de la revue épidémiologique du Nigéria (2022).

```
# Lecture du fichier shapefile
nga_adm1 <- 
  sf:::st_read(here::here("data/raw/NGA_adm_shapefile/NGA_adm1.shp")) %>%
  sf:::st_simplify(dTolerance = 1000) # simplifier le shapefile pour accélérer
  la visualisation.
```

```
## Reading layer `NGA_adm1' from data source
##
## /Users/kendavidn/Dropbox/tgc_github_projects/epi_reports_staging/data/raw/NGA_adm_shapefile
##   using driver `ESRI Shapefile'
## Simple feature collection with 38 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: 2.668431 ymin: 4.270418 xmax: 14.67642 ymax: 13.89201
## Geodetic CRS:  WGS 84
```

Les éléments importants de n'importe quelle couche ggplot sont les associations esthétiques `aes(x, y, ...)` qui indiquent à ggplot où placer les objets du graphique.

Nous pouvons imaginer une carte comme un graphique avec toutes les caractéristiques associées à un axe x et un axe y. Tous les types de géométrie (`geom_`) dans ggplot ont une sorte d'association esthétique, et celles-ci peuvent être déclarées au niveau du graphique, par exemple, `ggplot(data, aes(x = variable1, y = variable2))`, ou au niveau de la couche individuelle, par exemple, `geom_point(aes(color = variable3))`.

Ci-dessous, vous pouvez voir que `geom_sf()` est utilisé pour tracer les limites des différents États du Nigéria. De manière similaire, différentes couches peuvent être ajoutées par-dessus.

```
ggplot() +
  geom_sf(data = nga_adm1) +
  labs(title = "Aperçu du fichier shapefile du Nigéria")
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.
```

Ici, nous lisons d'abord le fichier shapefile du Nigéria en utilisant `sf::st_read()` puis nous le représentons à l'aide de `ggplot2`. La fonction `geom_sf()` est utilisée pour afficher les données spatiales, et `labs()` est utilisé pour ajouter un titre au graphique.



Étant donné que le fichier shapefile a été chargé à l'aide de `sf::st_read()`, nous n'avons pas besoin de spécifier les noms des axes. En fait, les coordonnées sont stockées en tant qu'objet multipolygone dans la variable de géométrie, et `geom_sf()` les reconnaît automatiquement.

Étape 2 : Importation des données attributaires



Dans le contexte des cartes choroplèthes, les "données attributaires" font référence aux informations quantitatives ou qualitatives qui seront utilisées pour ombrer ou colorer les différentes zones géographiques sur la carte. Par exemple, si vous avez une carte des régions d'un pays et que vous souhaitez ombrer chaque région en fonction de sa population, les données de population seront considérées comme des données attributaires.

Comme indiqué ci-dessus, nous utiliserons le nombre de cas de paludisme signalés au Nigéria pour les années 2000, 2006, 2010, 2015 et 2021.

```
# Lecture des données attributaires
malaria_cases <- read_csv(here::here("data/malaria.csv"))
malaria_cases
```

Étape 3 : Vérification des Données Jointes

Il est essentiel de vérifier et de valider les données jointes pour garantir le succès de la fusion et l'exactitude des données.

```
# Valider les données jointes
all.equal(unique(nga_adm1$NAME_1), unique(malaria_cases$state_name))
```

```
## [1] "Lengths (38, 37) differ (string compare on first 37)"
## [2] "2 string mismatches"
```

```
# Identifier les divergences
setdiff(unique(nga_adm1$NAME_1), unique(malaria_cases$state_name))
```

```
## [1] "Water body"
```

Dans l'extrait de code fourni, nous comparons les noms de région uniques entre les ensembles de données `nga_adm1` et `malaria_cases` en utilisant la fonction `all.equal()`. Cela vise à garantir que toutes les régions dans le fichier de formes correspondent à leurs homologues dans les données attributives.

Si les noms de région sont identiques, `all.equal()` renverra `TRUE`. Cependant, s'il y a des divergences, il détaillera les différences entre les deux ensembles de noms de région.

Il est à noter que "Water body" est présent dans le fichier de formes mais pas dans les `malaria_cases`. Étant donné que "Water body" n'est pas une région proprement dite, il doit être supprimé avant de fusionner les ensembles de données. Cela peut être fait avec la fonction `filter()`.

```
nga_adm1 <- filter(nga_adm1, NAME_1 != "Water body")
```

Étape 4 : Fusion des Données par Niveaux Administratifs

Maintenant, nous allons obtenir les données que nous souhaitons représenter sur la carte. L'élément essentiel est que les noms de région soient les mêmes dans le fichier de formes que dans les données que vous souhaitez représenter, car cela sera nécessaire pour les fusionner correctement.

Avant de fusionner les deux ensembles de données, nous devons définir la clé de fusion (`by =`).



REMINDER

N'oubliez pas de consulter la leçon sur la fusion de tables pour plus de détails sur la manière de fusionner des `data.frames` dans R si vous n'êtes pas familier avec la fusion.

```
# Ajouter les données de population et calculer les cas pour 10 000 habitants
malaria <- malaria_cases %>%
  left_join(nga_adm1,
            by = c("state_name" = "NAME_1")) %>%
  st_as_sf() # convertir en fichier de formes

# Sélectionner les colonnes les plus importantes
malaria2 <- malaria %>%
  select(state_name, cases_2000, cases_2006, cases_2010, cases_2015,
         cases_2021, geometry)
```

Dans cette étape, nous fusionnons les données du fichier de formes `nga_adm1` avec les données `malaria_cases` en utilisant `left_join()` du package `dplyr`. L'argument `by` est

utilisé pour spécifier la colonne commune sur laquelle fusionner les ensembles de données.

Enfin, nous convertissons les données fusionnées en un fichier de formes en utilisant `st_as_sf()`.

Nous pouvons conserver uniquement les variables importantes qui seront utilisées dans la construction des graphiques en utilisant `select()`.

RECAP



Dans cette partie de la leçon, nous avons appris l'importance des niveaux administratifs dans les données spatiales et comment fusionner les données spatiales et les données attributives par niveaux administratifs en utilisant le package `dplyr` de R, ainsi que comment vérifier et valider les données jointes.

Création d'une belle carte choroplète avec `ggplot2`

Utilisation de la variable de remplissage (c'est-à-dire la variable attributaire)

Pour afficher le nombre de cas, par exemple pour l'année 2021, nous devons remplir les polygones dessinés avec `geom_sf()` en utilisant la variable `fill`. Cela est très simple car il suit la même logique que la syntaxe du package `ggplot2`.

```
ggplot(data=malaria2) +  
  geom_sf(aes(fill=cases_2021)) +  # définir le remplissage en fonction de la  
  # variable de comptage des cas  
  labs(title = "Cas de Paludisme Distribués au Nigéria")
```

```
## Error in `calc_limits_bbox()`:  
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`  
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =  
## NULL`.
```

Nous créons une carte choroplète de base à l'aide de `ggplot2`. L'esthétique `fill` est définie pour varier en fonction de la variable `cases`, ce qui colore les régions en fonction du nombre de cas de paludisme.

Personnalisation de la carte

Nous pouvons personnaliser la carte en ajoutant des titres aux axes et à la légende, en ajoutant une flèche indiquant le nord et une échelle à l'aide de `ggspatial::annotation_north_arrow()` et `ggspatial::annotation_scale()`, et en changeant le thème en `theme_bw()`.

```

ggplot(data=malaria2) +
  geom_sf(aes(fill=cases_2021)) + # définir le remplissage en fonction de la
  # variable de comptage des cas
  labs(title = "Cas de Paludisme Distribués au Nigéria (2021)",
       fill = "Nombre de Cas") +
  xlab("Longitude") +
  ylab("Latitude") +
  ggspatial::annotation_north_arrow(location = "br") +
  ggspatial::annotation_scale(location = "bl") +
  theme_bw()

```

```

## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.

```



Dans cette section, nous avons d'abord créé une carte choroplète de base à l'aide de `ggplot2` pour visualiser les cas de paludisme régionaux au Nigéria pour 2021. Ensuite, nous avons amélioré la carte en ajoutant des titres, des étiquettes d'axes, une flèche indiquant le nord, une échelle, et en appliquant un thème monochrome.

1. Construisez une carte choroplète pour afficher la répartition des cas de paludisme en 2019, en utilisant la colonne `cases_2019` de l'ensemble de données `malaria2`. Vous pouvez améliorer la conception et la clarté de votre carte en incorporant des titres, des étiquettes d'axes et d'autres étiquettes pertinentes.

Échelle des couleurs

Échelle des couleurs pour les attributs continus

La fonction `scale_fill_continuous()` du package `ggplot2` en R est utilisée pour appliquer une échelle de couleurs continue à une carte choroplète.

Nous pouvons personnaliser la palette de couleurs utilisée pour l'échelle de couleurs continues en spécifiant les paramètres `low` et `high` dans la fonction `scale_fill_continuous()`.

```

# Créer un objet ggplot
ggplot(data = malaria2) +
  geom_sf(aes(fill = cases_2021)) +
  scale_fill_continuous(low = "green", high = "red") + # appliquer une échelle
  # de couleurs continue
  theme_bw()

```

```
## Error in `calc_limits_bbox()`:  
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`  
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =  
NULL`.
```



Dans cette section, nous avons appris comment appliquer une échelle de couleurs continue à une carte choroplète à l'aide de la fonction `scale_fill_continuous()` du package `ggplot2` en R et comment personnaliser la palette de couleurs.



Les données discrètes sont un type de données quantitatives qui ne peuvent prendre que des valeurs spécifiques et distinctes. Elles sont souvent le résultat du décompte d'objets ou d'événements. Les données discrètes sont importantes dans les cartes choroplèthes car elles nous permettent de représenter le décompte ou la quantité d'objets ou d'événements dans différentes régions.

La fonction `scale_fill_brewer()` du package `ggplot2` en R est utilisée pour appliquer une échelle de couleurs discrètes à une carte choroplète.

Avant d'appliquer l'échelle de couleurs discrètes, nous devrons créer une nouvelle colonne discrète, qui pourrait être un niveau de risque basé sur le nombre de cas. Pour cela, nous pouvons utiliser `mutate()` combinée avec `case_when()`.

```
# Transformation des données : création d'une nouvelle colonne 'risk' basée  
# sur le nombre de cas en 2021  
malaria2 %>%  
  mutate(risk = case_when(cases_2021 < quantile(cases_2021, 0.5) ~ 'faible',  
                          cases_2021 > quantile(cases_2021, 0.75) ~ 'élevé',  
                          TRUE ~ 'moyen')) -> malaria3  
  
# Visualisation des données  
ggplot(data = malaria3) +  
  geom_sf(aes(fill = fct_reorder(risk, cases_2021))) # Les niveaux de risque  
  # sont réorganisés en fonction du nombre de cas  
  scale_fill_brewer(palette = "Set4", "Niveaux de risque") +  
  theme_bw()
```

```
## Error in `calc_limits_bbox()`:  
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
```

```
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs = NULL`.
```

Vous pouvez également créer une palette de couleurs personnalisée pour les variables discrètes.

```
palette_personnalisee <- c("yellow", "orange", "red") # créer manuellement une palette de couleurs personnalisée

# Appliquer la palette de couleurs personnalisée
ggplot(data = malaria3) + # Créer un objet ggplot
  geom_sf(aes(fill = fct_reorder(risk, cases_2021)))+ # réorganiser les étiquettes de risque en fonction du nombre de cas
  scale_fill_manual(values = palette_personnalisee, "Niveaux de risque") +
  coord_sf(expand = TRUE) +
  theme_bw()
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs = NULL`.
```



RECAP Dans cette section, nous avons appris comment appliquer une échelle de couleurs discrètes à une carte choroplète à l'aide de la fonction `scale_fill_brewer()` du package `ggplot2` et comment personnaliser la palette de couleurs.



PRACTICE 2. Créez votre propre palette de couleurs distincte de celle initiale fournie ci-dessous, et affichez les cas de paludisme au Nigéria pour l'année 2000 en utilisant cette palette de couleurs personnalisée. N'oubliez pas d'incorporer des améliorations esthétiques supplémentaires.

Facet Wrap vs. Grid

- `facet_wrap()` : Cette fonction enveloppe une séquence 1D de panneaux en 2D. C'est utile lorsque vous avez une seule variable avec de nombreux niveaux et que vous souhaitez disposer les graphiques de manière plus économique en termes d'espace.
- `facet_grid()` : Cette fonction crée une matrice de panneaux définie par des variables de facettage en lignes et en colonnes. Elle est particulièrement utile

lorsque vous avez deux variables discrètes, et que toutes les combinaisons des variables existent dans les données.



- `facet_wrap()` est utilisé pour le facettement d'une seule variable, tandis que `facet_grid()` est utilisé pour le facettement de deux variables.
- `facet_wrap()` dispose les panneaux dans une grille 2D, tandis que `facet_grid()` les dispose dans une matrice.

Conclusion

Félicitations, vous avez terminé cette leçon !

Dans cette leçon, vous avez appris :

- Ce qu'est une carte choroplète et ses composantes.
- Les avantages et les limites des cartes choroplèthes.
- Comment préparer les données pour créer une carte choroplète.
- Comment créer une carte choroplète à l'aide de `ggplot2`.
- Comment appliquer des échelles de couleurs continues et discrètes.
- Comment créer des multiples avec `facet_wrap()` et `facet_grid()`.

Avec ces connaissances, vous êtes maintenant prêt à créer vos propres cartes choroplèthes à l'aide de `ggplot2` en R.

Dernières réflexions

La création de cartes choroplèthes est une compétence essentielle pour toute personne travaillant avec des données géographiques. Avec les outils et les techniques que vous avez appris dans cette leçon, vous pouvez maintenant créer vos propres cartes choroplèthes en utilisant `ggplot2` en R. N'hésitez pas à expérimenter avec différentes ensembles de données, palettes de couleurs et variables de facettement pour créer des cartes informatives et visuellement attrayantes. Bonne cartographie !

Références

1. Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis (3e)*. Available at: <https://ggplot2-book.org/>
2. Hadley Wickham and Garrett Grolemund. *R for Data Science*. Available at: <https://r4ds.had.co.nz/>
3. Robin Lovelace, Jakub Nowosad, and Jannes Muenchow. *Geocomputation with R*. Available at: <https://r.geocompx.org/>

Récapitulatif

Bravo d'être arrivé à la conclusion de la leçon d'aujourd'hui !

Tout au long de notre parcours aujourd'hui, nous avons exploré :

- Les concepts fondamentaux d'une carte choroplèthe et ses éléments essentiels.
- L'évaluation des avantages et des défis liés à l'utilisation des cartes choroplèthes.
- Les étapes pour organiser vos données de manière appropriée pour une carte choroplèthe.
- La maîtrise de l'art de créer une carte choroplèthe à l'aide de ggplot2.
- Les techniques pour appliquer des schémas de couleurs à la fois continus et discrets.
- Les méthodes pour produire de petits multiples en utilisant `facet_wrap()` et `facet_grid()`.

Vous êtes désormais armé de la compétence nécessaire pour créer des cartes choroplèthes distinctes en R à l'aide de ggplot2.

Solution

1.

```
ggplot(data=malaria2) +  
  geom_sf(aes(fill=cases_2015)) +  
  labs(title = "Cas de paludisme au Nigéria (2015)",  
       fill = "Nombre de cas",  
       x = "Longitude",  
       y = "Latitude") +  
  ggspatial::annotation_north_arrow(location = "br") +  
  ggspatial::annotation_scale(location = "bl") +  
  theme_classic()
```

```
## Error in `calc_limits_bbox()`:  
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`  
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =  
NULL`.
```

2.

```
new_palette <- c("grey", "blue", "red") # Définissez un ensemble de couleurs  
différent  
ggplot(data=malaria3) +  
  geom_sf(aes(fill = fct_reorder(risk, cases_2000))) + # Réorganisez les  
  étiquettes de risque en fonction des cas de l'année 2000  
  scale_fill_manual(values = new_palette, "Niveaux de risque pour l'année  
  2000") +  
  labs(title = "Distribution des cas de paludisme au Nigéria (2000)",  
    fill = "Nombre de cas",  
    x = "Longitude",  
    y = "Latitude") +  
  coord_sf(expand = TRUE) +  
  theme_minimal()
```

```
## Error in `calc_limits_bbox()`:  
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`  
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =  
NULL`.
```

3. Pour catégoriser les données en fonction de la médiane au lieu des quantiles, nous devons apporter une petite modification à la fonction `mutate()` où se produit la catégorisation du risque. Plus précisément, au lieu d'utiliser la fonction des quantiles, nous utiliserons la fonction de la médiane pour calculer la médiane des cas et diviser les données en risques “faibles” et “élevés” en fonction de cette valeur.

```

# Pivoter les données
malaria3_longer <- malaria2 %>%
  pivot_longer(cols = `cases_2000`:`cases_2021`, names_to = "year", values_to
  = "cases")

# Catégoriser le risque en fonction de la médiane
malaria3_longer %>%
  mutate(risk = case_when(
    cases <= median(cases) ~ 'faible',
    cases > median(cases) ~ 'élevé'
  )) -> malaria3_longer2

# Nettoyer les valeurs de l'année
malaria3_longer2$year <- str_replace(malaria3_longer2$year, "cases_", "")

# Tracer les données
ggplot() +
  geom_sf(data = nga_adm1) +
  geom_sf(aes(fill = cases), data = filter(malaria3_longer2, year %in%
    c("2000", "2021"))) +
  facet_grid(year ~ risk) +
  coord_sf(expand = TRUE) +
  scale_fill_continuous(low = "green", high = "red") +
  labs(title = "Distribution of Malaria cases and risk levels in Nigeria",
       subtitle = "2000 & 2021") +
  xlab("Longitude") +
  ylab("Latitude") +
  theme_bw()

```

```

## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()` 
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.

```

Contributeurs

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



IMAD EL BADISY

Data Science Education Officer
Deeply interested in health data



JOY VAZ

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

Amélioration des cartographies épidémiques avec des annotations

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

October 2023

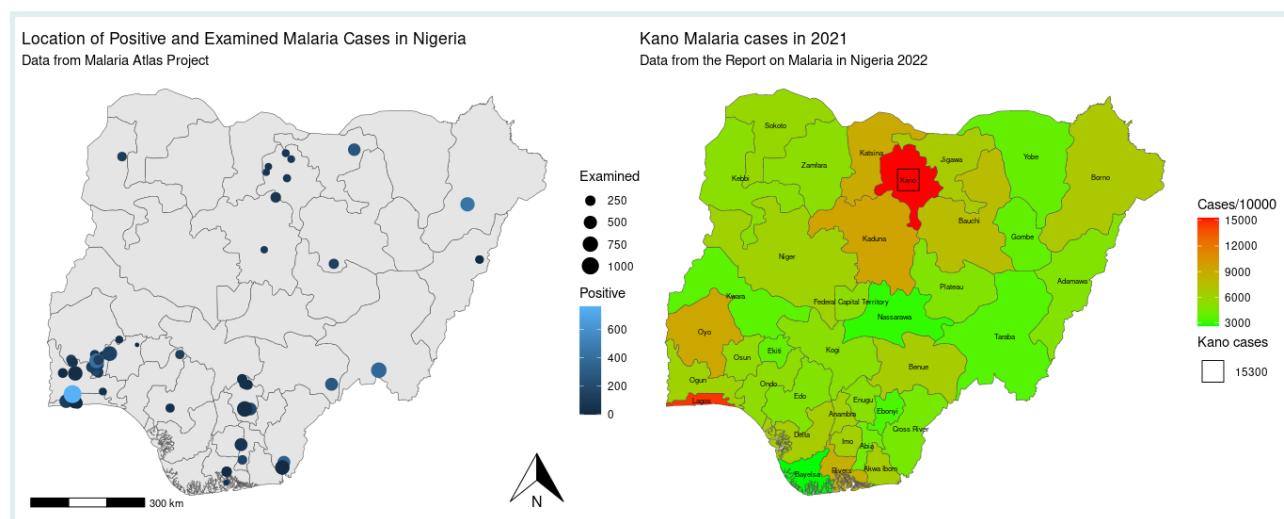
This document is a draft of a lesson made by the GRAPH Network, a non-profit headquartered at the University of Geneva Global Health Institute, in collaboration with the World Health Organization, under a Global Fund 2023 grant to create e-learning modules to build in-country data capacity for epidemiological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

Introduction
Packages
Préparation des Données
Création d'une Carte Choroplète Simple
Ajout d'Indicateurs de Données Continues à la Carte Choroplète
Exploration des Taux d'Augmentation des Cas de Paludisme à l'aide de Cartes Choroplèthes
Étiquetage de la Carte Choroplète avec les Noms des États
Affichage des Noms des États Combinés et des Taux d'Augmentation sur la Carte Choroplète
Mettre en Évidence une Région Spécifique sur la Carte tout en Préservant le Contexte
Étiquetage des Emplacements des Points : Exploration du Taux de Positivité au Paludisme et de l'Incidence
Dernières Réflexions
Références
Récapitulatif
Réponses

Introduction

Dans la visualisation des données géospatiales, les cartes sont des outils puissants pour raconter des histoires. Cependant, une carte sans annotations claires et des étiquettes équivaut à un livre sans titres ni chapitres. Bien que l'histoire puisse toujours être là, il devient considérablement plus difficile de comprendre, d'interpréter et d'apprécier.

Dans cette leçon, nous mettons particulièrement l'accent sur l'importance de l'annotation et de l'étiquetage des cartes. Une annotation appropriée transforme une visualisation simple en un guide informatif, permettant aux spectateurs de saisir rapidement des données spatiales complexes. Grâce à un étiquetage précis, les zones d'intérêt peuvent être immédiatement reconnues, facilitant une compréhension plus claire de la narration des données.



Objectifs d'apprentissage

Objectifs d'apprentissage : Techniques Avancées de Visualisation Géospatiale

À la fin de cette section, vous devriez être capable de :

1. Incorporer des indicateurs de données continues dans les cartes choroplèthes pour une granularité accrue.
2. Calculer et visualiser les taux d'augmentation des cas de paludisme à l'aide de cartes choroplèthes.
3. Superposer efficacement les noms des États sur les cartes choroplèthes, en veillant à la clarté et à la lisibilité.
4. Intégrer sans heurts les noms des États avec les taux d'augmentation sur les cartes sans compromettre la lisibilité.
5. Appliquer des techniques pour mettre en évidence des régions spécifiques sur une carte tout en conservant le contexte global.
6. Déterminer des stratégies optimales de placement des points et les intégrer efficacement dans les visualisations géospatiales.

En maîtrisant ces compétences, vous disposerez des connaissances et outils nécessaires pour créer des visualisations géospatiales riches, détaillées et informatives.

Packages

```
# Charger les bibliothèques
if(!require(pacman)) install.packages("pacman")
pacman::p_load(malariaAtlas,
  ggplot2,
  geodata,
  dplyr,
  here,
  readr,
  sf,
  patchwork)

# Désactiver la notation scientifique
options(scipen=100000)
```

Préparation des Données

Avant de plonger dans toute visualisation ou analyse, il est essentiel de charger et de prétraiter nos données. Cela comprend la lecture des datasets, la fusion des informations connexes et la filtration des entrées inutiles ou non pertinentes.

```
# Lecture des données géographiques au format shapefile
nga_adm1 <- sf::st_read(here::here("data/raw/NGA_adm_shapefile/NGA_adm1.shp"))
%>%
sf::st_simplify(dTolerance = 1000) # simplifier le shapefile pour accélérer
la visualisation.
```

```
## Reading layer `NGA_adm1` from data source
##
`/Users/kendavidn/Dropbox/tgc_github_projects/epi_reports_staging/data/raw/NGA_adm_shapefile'
##   using driver `ESRI Shapefile'
## Simple feature collection with 38 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: 2.668431 ymin: 4.270418 xmax: 14.67642 ymax: 13.89201
## Geodetic CRS:  WGS 84
```

Les données géographiques au format shapefile des limites administratives du Nigeria (comme les États ou les provinces) sont lues et stockées dans l'objet `nga_adm1`.

```
# Lecture des données attributives relatives aux cas de paludisme
malaria_cases <- read_csv(here::here("data/malaria.csv"))
```

Cette étape charge les données relatives aux cas de paludisme dans différents États du Nigeria.

```
# Filtrage des entrées 'Water body' des données géographiques
nga_adm1 <- filter(nga_adm1, NAME_1 != "Water body")

# Fusion des données géographiques avec les données sur le paludisme
malaria <- malaria_cases %>%
  left_join(nga_adm1, by = c("state_name" = "NAME_1")) %>%
  st_as_sf()
```

Ici, nous combinons les données sur les limites géographiques avec les données sur les cas de paludisme en utilisant les noms des États comme référence. Ces données fusionnées sont ensuite converties dans un format adapté aux visualisations géospatiales.

```
# Filtrage pour ne conserver que les colonnes essentielles pour notre analyse
malaria2 <- malaria %>%
  select(state_name, cases_2000, cases_2006, cases_2010, cases_2015,
cases_2021, geometry)
```

Nous réduisons notre dataset à des colonnes spécifiques, principalement les noms des États, les cas de paludisme de différentes années et les limites géographiques de ces États (c'est-à-dire `geometry`).

```
# Lecture des données de population pour différentes régions du Nigeria
population_nigeria <- read_csv(here::here("data/population_nigeria.csv"))
```

Cette étape charge les données indiquant la population des différents États ou régions du Nigeria.

```
# Combinaison des données de population avec nos données sur le paludisme
malaria3 <- malaria2 %>%
  left_join(population_nigeria, by = c("state_name")) %>%
  st_as_sf()
```

En fusionnant les données de population avec nos données sur les cas de paludisme, nous enrichissons notre dataset. Ces données combinées permettent des visualisations et des analyses plus complètes, telles que le calcul des taux d'incidence ou l'évaluation des tendances par rapport à la taille de la population.

Création d'une Carte Choroplète Simple

REMINDER



Les cartes choroplèthes sont des outils de visualisation puissants qui affichent des zones géographiques divisées ombrées ou hachurées en proportion de la valeur d'une variable.

Dans cet exemple, nous allons utiliser une carte choroplète pour visualiser la répartition des cas de paludisme dans différentes régions du Nigeria pour l'année 2021.

```
# Construction de la carte choroplète à l'aide de ggplot2
ggplot(data=malaria3) +
  geom_sf(aes(fill=cases_2021/1000)) + # La couleur de remplissage est
  # déterminée par le nombre de cas de paludisme en 2021, mis à l'échelle
  # par 1000
  labs(title = "Répartition des Cas de Paludisme au Nigeria en 2021", fill =
    "Cas pour 1000") + # Ajout de libellés et du titre à la carte
  scale_fill_continuous(low = "green", high = "red") + # Utilisation d'une
  # échelle de couleur continue allant du vert au rouge
  theme_void() # Utilisation d'un thème minimal pour une meilleure
  # visualisation de la carte
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.
```

Voici une explication détaillée du code :

- `ggplot(data=malaria3)`: Initialise un objet ggplot en utilisant le dataset malaria3.

- `geom_sf(aes(fill=cases_2021/1000))`: Ajoute les données géographiques de `malaria3` et remplit chaque région en fonction du nombre de cas de paludisme en 2021, mis à l'échelle par un facteur de 1000. Cela représente efficacement le nombre de cas pour mille habitants.
- `labs(title = "Répartition des Cas de Paludisme au Nigeria en 2021", fill = "Cas pour 1000")`: Spécifie le titre de la carte et l'étiquette de l'échelle de couleur.
- `scale_fill_continuous(low = "green", high = "red")`: Applique une échelle de couleur continue où les régions avec moins de cas sont colorées en vert et les régions avec plus de cas sont colorées en rouge.
- `theme_void()`: Supprime le texte des axes, les marques de graduations et autres éléments non essentiels pour mettre en évidence la carte.

Le graphique résultant offre une vue claire de la répartition des cas de paludisme au Nigeria en 2021, avec l'intensité des couleurs indiquant l'ampleur des cas dans chaque région.

Exercice 1 : Modifiez la carte choroplèthe fournie pour visualiser la répartition des cas de paludisme au Nigeria pour l'année 2015

Instructions

1. Mettez à jour la correspondance des données dans la fonction `geom_sf()` pour refléter les cas de paludisme pour l'année 2015.
2. Ajustez le titre dans la fonction `labs()` pour indiquer que la visualisation concerne l'année 2015.
3. Modifiez le dégradé de couleurs dans `scale_fill_continuous()` pour passer du bleu (cas "faibles") au jaune (cas "elevés").

Ci-dessous, un code de départ :



```
# Construction de la carte choroplèthe pour 2015 à l'aide de
# ggplot2
ggplot(data=malaria3) +
  geom_sf(aes(fill=_____)) + # Remplissez la colonne de
  # données correcte pour 2015
  labs(title = "_____ ", fill = "Cas pour
  1000") + # Mettez à jour le titre de manière appropriée
  scale_fill_continuous(_____) + # Modifiez l'échelle de
  # couleurs
  theme_void()
```

Ajout d'Indicateurs de Données Continues à la Carte Choroplète

Lors de l'analyse de données sur les maladies, il est souvent utile de regarder au-delà des chiffres bruts de cas et de se concentrer sur les taux, en particulier les taux d'incidence. Le taux d'incidence fournit une mesure normalisée qui peut prendre en compte les différences de taille de population entre les régions, rendant les comparaisons plus significatives.

Compréhension de l'Incidence

L'incidence d'une maladie est un pilier de la recherche épidémiologique. Elle quantifie le nombre de nouveaux cas d'une maladie survenant au cours d'une période spécifique, généralement une année, par rapport à une population à risque.

Mathématiquement, il est donné par :

$$\text{Incidence} = \frac{\text{Nombre de nouveaux cas au cours de la période}}{\text{Population à risque au début de la période}}$$

Dans ce contexte, nous examinons l'incidence du paludisme dans différents États du Nigeria pour l'année 2021. Plus précisément, nous calculerons le taux d'incidence en divisant le nombre de nouveaux cas de paludisme en 2021 par la population de chaque État issue du dernier recensement disponible en 2019.

Le code R suivant réalise cela et visualise les données :

```
# Trouver les coordonnées du centre de chaque État pour positionner les étiquettes
centroid_coords <- st_coordinates(st_centroid(malaria2$geometry))

# Visualisation du Taux d'Incidence du Paludisme en 2021 à l'aide d'une Carte Choroplète
ggplot(data = malaria3) +
  # Remplir chaque État avec une couleur représentant le taux d'incidence en 2021.
  geom_sf(aes(fill = round(cases_2021/population_2019, 2))) +
  # Étiqueter chaque État avec son taux d'incidence spécifique.
  geom_text(aes(x = centroid_coords[, 1], y = centroid_coords[, 2], label =
    round(cases_2021/population_2019, 2)), size = 2) +
  # Ajout de titre et de titre de légende.
  labs(title = "Taux d'Incidence du Paludisme au Nigeria en 2021", fill =
    "Incidence") +
  # Utilisation d'une échelle de couleur continue du vert (faible incidence) au rouge (forte incidence).
  scale_fill_continuous(low = "green", high = "red") +
  # Utilisation d'un thème minimaliste pour une visualisation plus claire.
  theme_void()
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
```

```
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =  
NULL`.
```

Voici une brève explication de la visualisation :

- La fonction `geom_sf()` crée une carte choroplète où la couleur de chaque État représente son taux d'incidence du paludisme en 2021.
- La fonction `geom_text()` étiquète chaque État avec son taux d'incidence spécifique, positionnant chaque étiquette au centre de l'État.
- L'échelle de couleurs, passant du vert au rouge, met en évidence visuellement les régions présentant des taux d'incidence plus élevés.

Cette visualisation offre une compréhension immédiate de la situation du paludisme au Nigeria, révélant les zones à forte incidence qui pourraient nécessiter des interventions de santé publique plus ciblées.

Exploration des Taux d'Augmentation des Cas de Paludisme à l'aide de Cartes Choroplèthes

Comprendre l'évolution du nombre de cas de maladies au fil du temps peut fournir des informations sur l'efficacité des interventions, la progression de la maladie et les zones où des ressources accrues pourraient être nécessaires. Dans ce contexte, nous cherchons à visualiser l'augmentation en pourcentage des cas de paludisme de 2015 à 2021 dans différents États du Nigeria.

Calcul du Taux d'Augmentation

Le taux d'augmentation pour chaque État est calculé comme suit :

$$\text{Taux d'Augmentation} = \left(\frac{\text{Cas en 2021} - \text{Cas en 2015}}{\text{Cas en 2015}} \right) \times 100\%$$

Cette formule fournit la croissance en pourcentage (ou la diminution) des cas de paludisme de 2015 à 2021.

Visualisation des Taux d'Augmentation

Plongeons dans le code qui réalise cette visualisation :

```

# Calculer le taux d'augmentation pour chaque État
malaria3 %>%
  mutate(increase_rate = round(((cases_2021 - cases_2015) / cases_2015) *
    100)) -> malaria3

# Visualisation des taux d'augmentation à l'aide d'une carte choroplète
ggplot(data = malaria3) +
  # Colorer chaque État en fonction de son taux d'augmentation
  geom_sf(aes(fill = increase_rate), color="white", size = 0.2) +
  # Utilisation d'une échelle de couleur continue pour représenter les taux
  # d'augmentation, passant du vert au rouge
  scale_fill_continuous(name="Taux d'augmentation (%)", limits=c(0,70), low =
    "green", high = "red", breaks=c(0, 20, 40, 60))+

  # Étiquetage de chaque État avec son taux d'augmentation en pourcentage
  geom_text(aes(x = centroid_coords[, 1], y = centroid_coords[, 2], label =
    paste0(increase_rate, "%")), size = 2)+

  # Ajout d'un titre à la carte
  labs(title = "Taux d'Augmentation du Paludisme au Nigeria en 2021 par
    rapport à 2015")+
  theme_void()

```

```

## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.

```

Dans cette visualisation :

- La fonction `geom_sf()` crée la carte choroplète où l'intensité de couleur de chaque État représente son taux d'augmentation du paludisme.
- `scale_fill_continuous()` définit l'échelle de couleur pour les taux d'augmentation, mettant en évidence davantage les zones à forte augmentation.
- `geom_text()` ajoute des étiquettes à chaque État, fournissant des valeurs exactes en pourcentage.
- La carte résultante nous permet d'identifier rapidement les régions avec une forte croissance des cas de paludisme sur la période sélectionnée.

Grâce à cette visualisation, les parties prenantes peuvent cibler les régions où le paludisme est en augmentation et potentiellement allouer plus efficacement des ressources.

Étiquetage de la Carte Choroplète avec les Noms des États

Dans une carte choroplète, bien que les dégradés de couleurs offrent un indice visuel pour comprendre la répartition d'une variable dans différentes régions, l'ajout d'étiquettes peut considérablement améliorer la clarté de la visualisation. Cela est particulièrement vrai lorsque le public n'est peut-être pas familier avec toutes les limites géographiques affichées. Dans le cas de notre dataset sur le paludisme, l'ajout des noms des États à la carte rend les données plus accessibles et compréhensibles.

Analysons le code :

```
# Construction d'une carte choroplète avec les noms des États
ggplot(data = malaria3) +
  # Remplir chaque État en fonction du nombre de cas de paludisme en 2021,
  # normalisé par 10 000
  geom_sf(aes(fill = cases_2021/10000)) +
  # Ajouter le nom de chaque État à son centre
  geom_text(aes(x = centroid_coords[, 1], y = centroid_coords[, 2], label =
    state_name), size = 2, check_overlap = TRUE) +
  # Ajouter des titres et des étiquettes
  labs(title = "Cas de Paludisme au Nigeria en 2021", fill = "Cas/10 000") +
  # Utiliser une échelle de couleur continue du vert (faible nombre de cas) au
  # rouge (nombre élevé de cas)
  scale_fill_continuous(low = "green", high = "red") +
  theme_void()
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.
```

Voici une explication détaillée de la visualisation :

- `geom_sf(aes(fill = cases_2021/10000))` : Cela crée la carte choroplète où la couleur de chaque État représente le nombre de cas de paludisme en 2021, normalisé par un facteur de 10 000.
- `geom_text(aes(...))` : Cette fonction place du texte sur le graphique. Dans ce cas, elle est utilisée pour ajouter le nom de chaque État à son centre géographique. Le paramètre `check_overlap = TRUE` permet à ggplot de vérifier les chevauchements de libellés de texte et de tenter d'éviter les chevauchements.
- `labs(title = "Cas de Paludisme au Nigeria en 2021", fill = "Cas/10 000")` : Cette fonction ajoute un titre au graphique et un libellé à l'échelle de couleur.
- `scale_fill_continuous(low = "green", high = "red")` : Cela définit l'échelle de couleur de la carte, en transitionnant du vert pour les États avec moins de cas au rouge pour ceux avec plus de cas.

La visualisation résultante est une carte claire et informative des cas de paludisme au Nigeria en 2021, avec chaque État étiqueté pour une référence facile.

Affichage des Noms des États Combinés et des Taux d'Augmentation sur la Carte Choroplèthe

Les visualisations peuvent transmettre une multitude d'informations lorsqu'elles intègrent de multiples points de données de manière intuitive. En associant les noms des États à leurs taux d'augmentation respectifs, nous pouvons fournir une vue plus riche et plus détaillée des données sans submerger le public.

Plongeons dans cette visualisation :

Nous souhaitons présenter une carte choroplèthe montrant les cas de paludisme pour 10 000 habitants dans les États nigérians en 2021, avec des étiquettes combinant les noms des États et leurs taux d'augmentation respectifs depuis 2015.

```
# Combinez les noms des États et leurs taux d'augmentation respectifs en une
# seule étiquette
malaria3$label_text <- paste(malaria3$state_name, malaria3$increase_rate, "%")

# Calculez le centre de chaque État pour servir de points de référence pour
# les étiquettes
centroid_coords <- st_coordinates(st_centroid(malaria3$geometry))

# Visualisez les données
ggplot(data = malaria3) +
  # Créez une carte choroplèthe ombrée en fonction du nombre de cas de
  # paludisme en 2021 pour 10 000 habitants
  geom_sf(aes(fill = cases_2021/10000)) +
  # Ajoutez des étiquettes combinées (nom de l'État et taux d'augmentation) au
  # centre de chaque État
  geom_text(aes(x = centroid_coords[, 1],
                y = centroid_coords[, 2],
                label = label_text), size = 2) +
  # Ajoutez des titres et personnalisez la légende des couleurs
  labs(title = "Cas de Paludisme au Nigeria en 2021", fill = "Cas/10 000") +
  scale_fill_continuous(low = "green", high = "red") +  # Définir la gradation
  # des couleurs
  theme_void()  # Appliquez un thème minimal pour plus de clarté
```

```
## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.
```

Dans cette visualisation :

- La ligne `malaria3$label_text <- paste(malaria3$state_name, malaria3$increase_rate, "%")` construit nos étiquettes combinées en concaténant le nom de l'État avec son taux d'augmentation, suivi d'un signe de pourcentage.

- `geom_text()` ajoute ces étiquettes combinées au graphique. Il positionne chaque étiquette au centre de l'État correspondant, en veillant à ce que les étiquettes soient centrées et facilement associées à leurs régions respectives.
- `scale_fill_continuous(low = "green", high = "red")` attribue une gradation de couleurs en fonction du nombre de cas de paludisme. Les États avec moins de cas seront colorés en vert, passant au rouge pour les États avec plus de cas.

Cette approche nous permet de communiquer efficacement deux informations essentielles (cas pour 10 000 habitants et taux d'augmentation) dans la même visualisation tout en conservant le nom des États pour une identification aisée.

Mettre en Évidence une Région Spécifique sur la Carte tout en Préservant le Contexte

Parfois, vous souhaitez attirer l'attention sur une zone ou une région particulière de votre carte sans omettre les détails des régions environnantes. En utilisant des éléments graphiques spécifiques, tels que des marqueurs plus grands ou des couleurs distinctes, vous pouvez mettre l'accent sur certaines régions tout en mettant en valeur les données plus larges. Dans cet exemple, nous mettons l'accent sur la région de "Kano" au Nigéria.

```

# Calcul des coordonnées du centroïde pour l'étiquetage
centroid_coords <- st_coordinates(st_centroid(malaria3$geometry))

# Visualisation des cas de paludisme au Nigeria avec une mise en avant de Kano
ggplot(data = malaria3) +
  # Création d'une carte choroplète avec des couleurs basées sur les cas de
  # paludisme en 2021
  geom_sf(aes(fill = cases_2021/1000)) +
  # Définition d'une échelle de couleurs continues du vert au rouge
  scale_fill_continuous(low = "green", high = "red") +
  # Superposition d'un point sur Kano pour le mettre en avant. La taille du
  # point correspond au nombre de cas
  geom_point(data = subset(malaria3, state_name == "Kano"),
             aes(x = st_coordinates(st_centroid(geometry))[1],
                  y = st_coordinates(st_centroid(geometry))[2], size =
round(cases_2021/1000)),
             color = "black", shape = 22, fill = "transparent") +
  # Étiquetage de chaque région avec son nom
  geom_text(aes(x = centroid_coords[, 1], y = centroid_coords[, 2], label =
state_name), size = 2, check_overlap = TRUE) +
  # Personnalisation de l'échelle de taille du point mis en avant
  scale_size_continuous(range = c(1, 12)) +
  # Ajout du titre et des légendes
  labs(title = "Cas de paludisme à Kano en 2021", subtitle = "Données du
Rapport sur le Paludisme au Nigeria 2022", size = "Cas à Kano", fill =
"Cas/10 000",) +
  # Application d'un thème minimal pour plus de clarté
  theme_void()

```

```

## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL`.

```

Dans cette visualisation :

- La fonction `geom_point()` est utilisée pour placer un cercle sur la région de Kano. La taille du cercle représente le nombre de cas à Kano en 2021. Le cercle est conçu en mode transparent (`fill = "transparent"`) avec une bordure noire audacieuse (`color = "black"`) pour le mettre en évidence.
- `geom_text()` ajoute les noms de toutes les régions sur la carte. Il place chaque étiquette près du centre géographique de la région respective. Le paramètre `check_overlap = TRUE` empêche les étiquettes de se chevaucher autant que possible, garantissant ainsi la lisibilité.



Tandis que "Kano" est mis en évidence, toutes les autres régions sont également affichées avec leur ombrage de couleur respectif basé sur les cas de paludisme. Cela offre une vue holistique de la situation à travers le Nigéria, permettant aux téléspectateurs de comparer Kano avec les autres régions.

Une telle approche est inestimable lorsque vous souhaitez mettre en lumière des détails spécifiques ou des zones d'intérêt sans négliger le jeu de données plus large, enrichissant ainsi vos présentations de données.

Étiquetage des Emplacements des Points : Exploration du Taux de Positivité au Paludisme et de l'Incidence

La cartographie et la visualisation de données spécifiques sur une carte géographique peuvent fournir des informations cruciales, notamment lors de la manipulation de données épidémiologiques. Plongeons dans le code pour comprendre les processus et la visualisation que nous cherchons à réaliser :

```
# Récupération des Données
# Le package malariaAtlas fournit la fonction `getPR` pour accéder au taux de
# parasite (PR).
# Le PR est un indicateur essentiel de la prévalence du paludisme.

# Récupération des données pour le Nigéria pour les deux espèces de paludisme
nigeria_pr <- malariaAtlas::getPR(ISO = "NGA", species = "both") %>%
  # Filtrer les enregistrements sans valeurs de PR
  filter(!is.na(pr)) %>%
  # Supprimer toutes les lignes avec des valeurs manquantes de longitude ou de
  # latitude
  drop_na(longitude, latitude) %>%
  # Convertir les données en un dataframe spatial pour faciliter la
  # cartographie
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326)
```

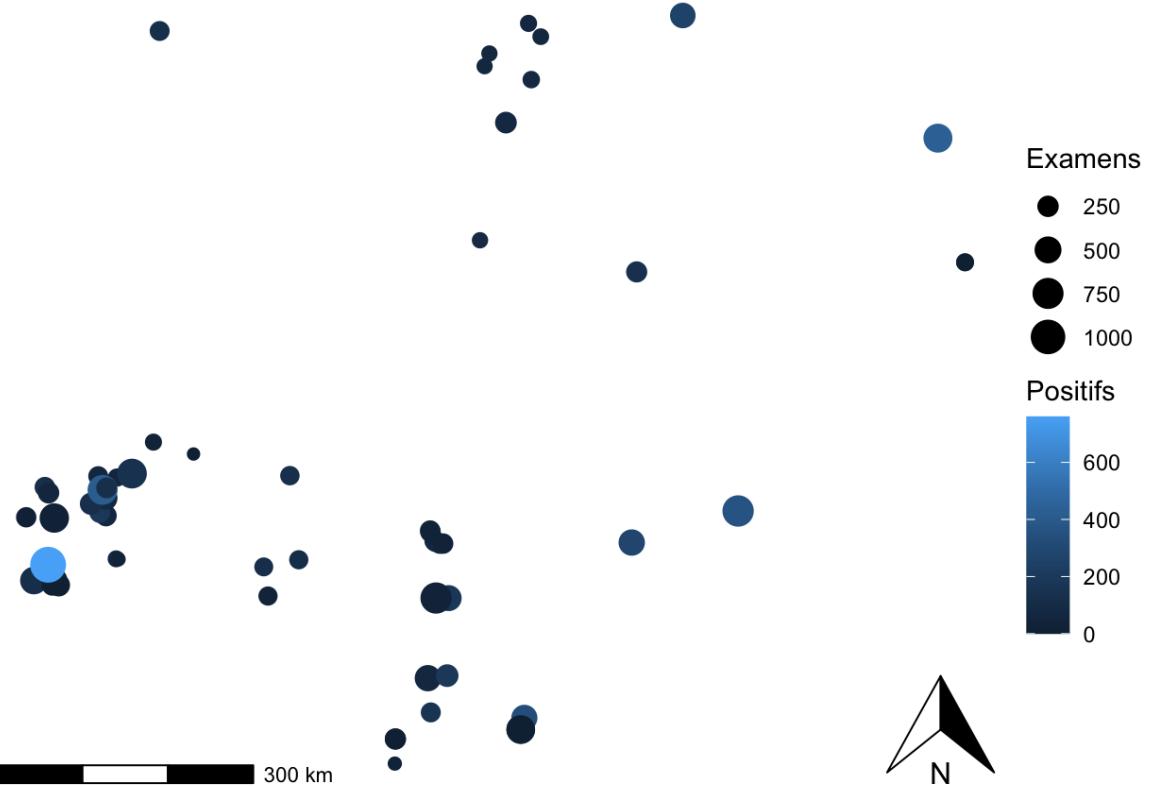
Ce bloc de code récupère les données de prévalence du paludisme pour le Nigéria. Après la récupération, les données sont nettoyées en filtrant les valeurs manquantes. Ensuite, elles sont transformées en un format adapté à la visualisation géospatiale (objet `sf`).

```

# Configuration d'une visualisation géospatiale avec ggplot2
ggplot() +
  # Traçage des limites administratives du Nigeria
  geom_sf(data = nga_adm1) +
  # Ajout de points pour chaque site de test
  # La couleur de chaque point indique si le test est positif, et la taille
  # représente le nombre de personnes testées
  geom_sf(aes(size = examined, color = positive), data = nigeria_pr) +
  # Définition des titres, des étiquettes des axes et des légendes du
  # graphique
  labs(title = "Emplacement des cas de paludisme positifs et examinés au
    Nigeria",
    subtitle = "Données du Projet Atlas du Paludisme",
    color = "Positifs",
    size = "Examens") +
  # Ajout d'étiquettes pour la longitude et la latitude
  xlab("Longitude") +
  ylab("Latitude") +
  # Intégration d'une flèche indiquant le nord pour l'orientation
  ggspatial::annotation_north_arrow(location = "br") +
  # Intégration d'une barre d'échelle pour aider à l'interprétation des
  # distances
  ggspatial::annotation_scale(location = "bl") +
  # Application d'un thème minimaliste pour une clarté visuelle
  theme_void()

```

Emplacement des cas de paludisme positifs et examinés au Nigeria
 Données du Projet Atlas du Paludisme





Ce bloc de code visualise les données sur le paludisme sur une carte. Il met en évidence les régions en fonction du nombre de tests de dépistage du paludisme et de leurs résultats. Des outils spécifiques du package `ggspatial` sont utilisés pour ajouter des éléments cartographiques, garantissant que la carte est informative.

Exercice 3 : Réfléchir sur le taux de positivité par la taille et la couleur



Créez une nouvelle visualisation qui représente le taux de positivité (nombre de cas positifs divisé par le nombre d'examens) pour chaque emplacement. Modifiez la taille des points pour refléter le taux de positivité.

Dernières Réflexions

La visualisation des données géospatiales, c'est bien plus que de représenter des données sur une carte. C'est raconter une histoire ancrée dans un lieu et un espace. Cette leçon s'est plongée dans une approche narrative basée sur les couches, de l'importance d'annotations claires à l'intégration de différents types de données pour une compréhension plus approfondie.

Références

1. Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis (3e)*. Available at: <https://ggplot2-book.org/>
2. Hadley Wickham and Garrett Grolemund. *R for Data Science*. Available at: <https://r4ds.had.co.nz/>
3. Robin Lovelace, Jakub Nowosad, and Jannes Muenchow. *Geocomputation with R*. Available at: <https://r.geocompx.org/>

Récapitulatif

Dans cette leçon sur l'étiquetage des cartes, nous avons exploré en profondeur les points suivants :



RECAP



- **Annotations et Étiquettes** : Des annotations claires transforment les cartes, rendant les données spatiales complexes facilement compréhensibles.
- **Techniques de Cartographie Avancées** : Nous avons exploré comment intégrer des indicateurs continus, superposer des étiquettes, mettre l'accent sur des régions et déterminer les placements optimaux des points sur les cartes.
- **Application Pratique** : En utilisant un dataset sur le paludisme, nous avons démontré la préparation des données, la visualisation et l'interprétation dans R.
- **Packages R** : Des packages tels que ggplot2, sf et ggspatial facilitent les visualisations géospatiales avancées.
- **Mettre en Évidence avec le Contexte** : Il est essentiel de fournir une vue d'ensemble, même lorsque l'on met l'accent sur des zones spécifiques.

Réponses

Solution pour l'exercice 1

Pour visualiser la répartition des cas de paludisme au Nigéria en 2015, suivez les instructions fournies dans l'exercice. Voici la solution complète :

```
# Construction de la carte choroplète pour 2015 avec ggplot2
ggplot(data=malaria3) +
  geom_sf(aes(fill=cases_2015/1000)) + # Colonne de données mise à jour pour
  # refléter l'année 2015
  labs(title = "Répartition des cas de paludisme au Nigéria en 2015", fill =
    "Cas pour 1000") + # Titre mis à jour pour 2015
  scale_fill_continuous(low = "blue", high = "yellow") + # Échelle de couleurs
  # modifiée pour un dégradé de bleu à jaune
  theme_void()
```

Solution pour l'exercice 2

Pour combiner la visualisation des taux d'augmentation avec les noms des états, suivez les étapes ci-dessous :

```

# Combinaison de la visualisation des taux d'augmentation avec les noms des états
ggplot(data = malaria3) +
  # Création d'une carte choroplète avec une couleur de remplissage basée sur le taux d'augmentation
  geom_sf(aes(fill = increase_rate), color="white", size = 0.2) +
  # Étiquetage de chaque région avec son nom d'état
  geom_text(aes(x = centroid_coords[, 1],
                y = centroid_coords[, 2],
                label = state_name), size = 2, check_overlap = TRUE) +
  # Spécification de l'échelle de couleurs pour les taux d'augmentation
  scale_fill_continuous(name="Taux d'augmentation (%)",
                        limits=c(0,70),
                        low = "green",
                        high = "red",
                        breaks=c(0, 20, 40, 60)) +
  # Ajout d'un titre et d'une légende à la carte
  labs(title = "Taux d'augmentation du paludisme au Nigéria de 2015 à 2021",
       fill = "Taux d'augmentation (%)") +
  theme_void()  # Application d'un thème minimaliste pour la clarté

```

```

## Error in `calc_limits_bbox()`:
## ! Scale limits cannot be mapped onto spatial coordinates in `coord_sf()`
## i Consider setting `lims_method = "geometry_bbox"` or `default_crs =
NULL` .

```

Dans cette solution, la carte choroplète est créée en fonction du taux d'augmentation des cas de paludisme de 2015 à 2021. Chaque état du Nigéria est étiqueté par son nom, et le dégradé de couleurs (du vert au rouge) met en évidence l'ampleur du taux d'augmentation. La carte est enrichie d'un titre et d'une légende pour assurer la clarté et la compréhension.

Solution pour l'exercice 3

Créez une visualisation qui représente le taux de positivité :

```

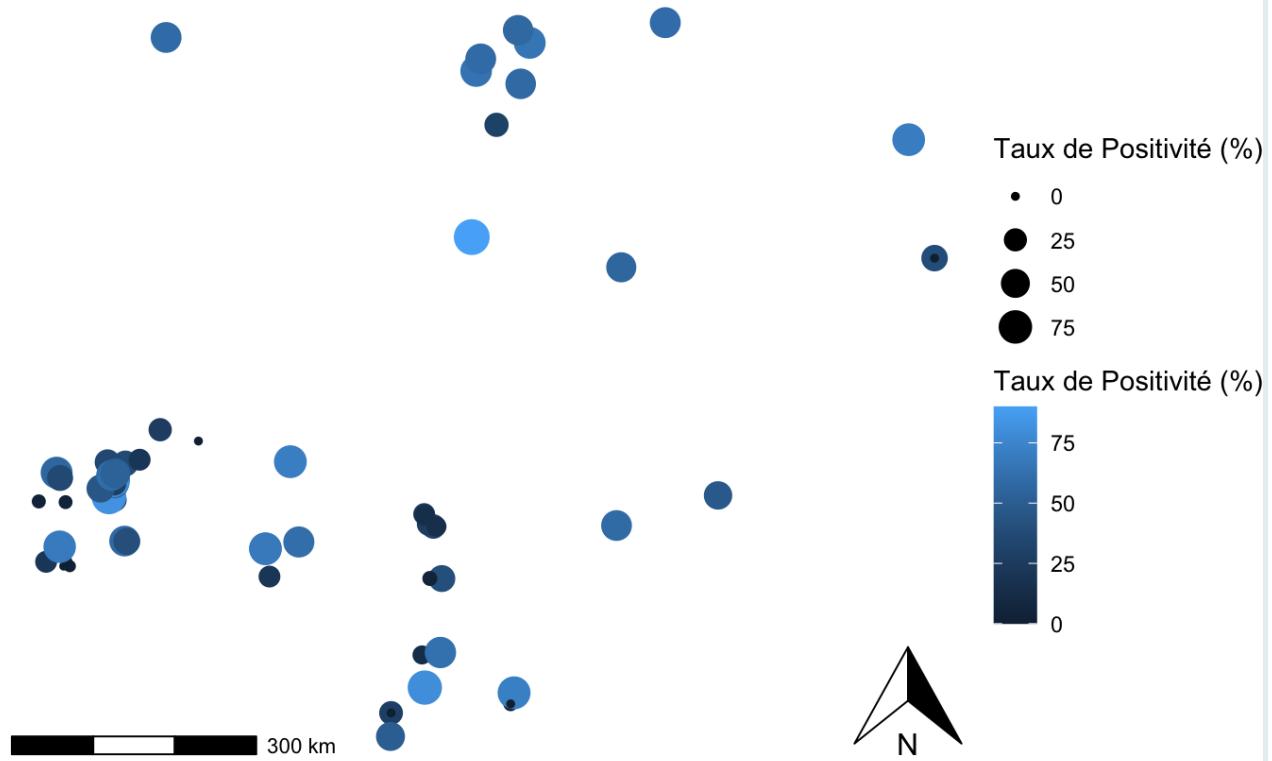
# Ajout d'une colonne de taux de positivité
nigeria_pr$positive_rate <- (nigeria_pr$positive / nigeria_pr$examined) * 100

ggplot() +
  geom_sf(data = nga_adm1) +
  geom_sf(aes(size = positive_rate, color = positive_rate), data =
    nigeria_pr) +
  labs(title = "Emplacement et Taux de Positivité des Cas de Paludisme au Nigéria",
       subtitle = "Données du Projet Atlas du Paludisme",
       color = "Taux de Positivité (%)",
       size = "Taux de Positivité (%)") +
  xlab("Longitude") +
  ylab("Latitude") +
  ggspatial::annotation_north_arrow(location = "br") +
  ggspatial::annotation_scale(location = "bl") +
  theme_void()

```

Emplacement et Taux de Positivité des Cas de Paludisme au Nigéria

Données du Projet Atlas du Paludisme



Contributeurs

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



IMAD EL BADISY

Data Science Education Officer

Deeply interested in health data



JOY VAZ

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