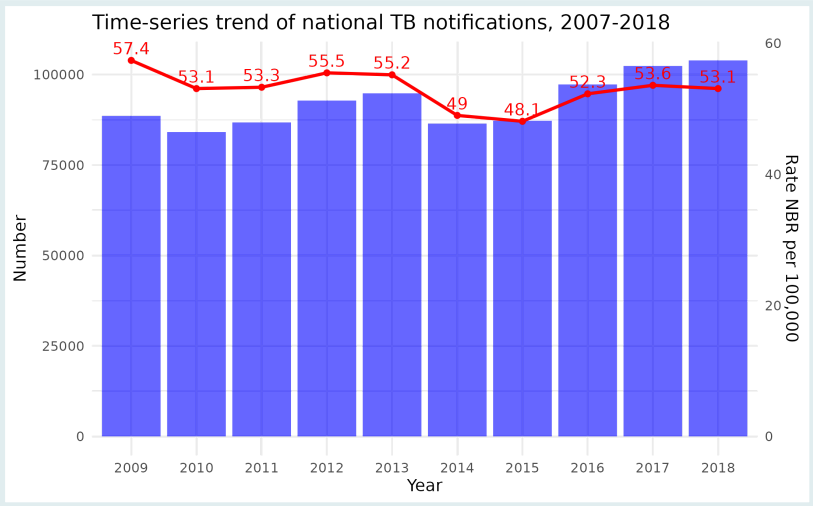
Visualisation de Séries Temporelles Épidémiologiques

Introduction

En analysant des données de séries temporelles, des données collectées à intervalles réguliers, les épidémiologistes peuvent générer des prévisions, informer les décisions politiques et, finalement, améliorer les mesures de prévention et de contrôle des maladies. Dans cette leçon, nous explorons comment créer et interpréter des séries temporelles épidémiologiques, en utilisant R pour visualiser les données de manière efficace.



Objectifs d'Apprentissage

À la fin de cette leçon, vous serez capable de :

- Remodeler les données de séries temporelles pour la représentation graphique avec pivot_longer()
- Créer des graphiques linéaires dans ggplot2 en associant le temps à l'axe x et les valeurs à l'axe y
- Améliorer l'esthétique des graphiques linéaires avec des techniques telles que les étiquettes personnalisées, les palettes de couleurs, les annotations
- Visualiser les intervalles de confiance avec geom_ribbon()
- Mettre en évidence les modèles dans des données bruyantes en utilisant le lissage et l'agrégation
- Comparer des séries temporelles avec des échelles distinctes en utilisant des axes doubles et sec_axis()

Packages

Installez et chargez les paquets nécessaires avec le bloc de code suivant :

```
# Charger les paquets
if(!require(pacman)) install.packages("pacman")
pacman::p_load(dplyr, ggplot2, tidyr, lubridate, outbreaks, scales, ggrepel,
         ggthemes, zoo, here)
options(scipen=999)
```

PRO TIP

Définir `options(scipen = 999)` empêche l'utilisation de la notation scientifique dans nos graphiques, rendant les nombres longs plus faciles à lire et à interpréter.

Introduction aux Graphiques Linéaires pour les Données de Séries Temporelles

Pour commencer à visualiser les données de séries temporelles, nous allons examiner la dynamique des notifications de tuberculose (TB) en Australie au fil du temps, en comparant les notifications dans les zones urbaines et rurales. Le jeu de données source est accessible ici

Notifications est un terme technique pour le nombre de cas d'une maladie qui sont signalés aux autorités de santé publique.

Préparation des Données : Agrégation et Pivotement

Commençons par charger et inspecter les données :

```
tb_data_aus <- read_csv(here::here("data/aus_tb_notifs.csv"))
head(tb_data_aus)
```

```
## # A tibble: 6 × 3
##   period rural urban
##   <chr>  <dbl> <dbl>
## 1 1993Q1     6    51
## 2 1993Q2    11    52
## 3 1993Q3    13    67
## 4 1993Q4    14    82
## 5 1994Q1    16    63
## 6 1994Q2    15    65
```

Ce jeu de données comprend les colonnes `period` (temps en format trimestriel, par exemple, '1993Q1'), `rural` (cas dans les zones rurales) et `urban` (cas dans les zones urbaines).

Nous aimerions visualiser le nombre de notifications annuelles de TB dans les zones urbaines et rurales, mais les données sont actuellement au format trimestriel. Ainsi, nous devons agréger les données par année.

Commençons par extraire l'année de la colonne `period`. Nous faisons cela en utilisant la fonction `str_sub()` du paquet `stringr` :

```r
tb_data_aus %>%
  mutate(year = str_sub(period, 1, 4)) %>%
  # reconvertir en numérique
  mutate(year = as.numeric(year))
```

```
## # A tibble: 120 × 4
##    period rural urban  year
##    <chr>  <dbl> <dbl> <dbl>
##  1 1993Q1     6    51  1993
##  2 1993Q2    11    52  1993
##  3 1993Q3    13    67  1993
##  4 1993Q4    14    82  1993
##  5 1994Q1    16    63  1994
##  6 1994Q2    15    65  1994
##  7 1994Q3    18    60  1994
##  8 1994Q4    25    71  1994
##  9 1995Q1     7    77  1995
## 10 1995Q2     9    52  1995
## # ℹ 110 more rows
```

La fonction `str_sub()` prend trois arguments : la chaîne à partir de laquelle nous voulons extraire, la position de départ et la position de fin. Dans ce cas, nous voulons extraire les quatre premiers caractères de la colonne `period`, qui correspondent à l'année.

Maintenant, agrégeons les données par année. Nous pouvons le faire en utilisant les fonctions `group_by()` et `summarise()` :

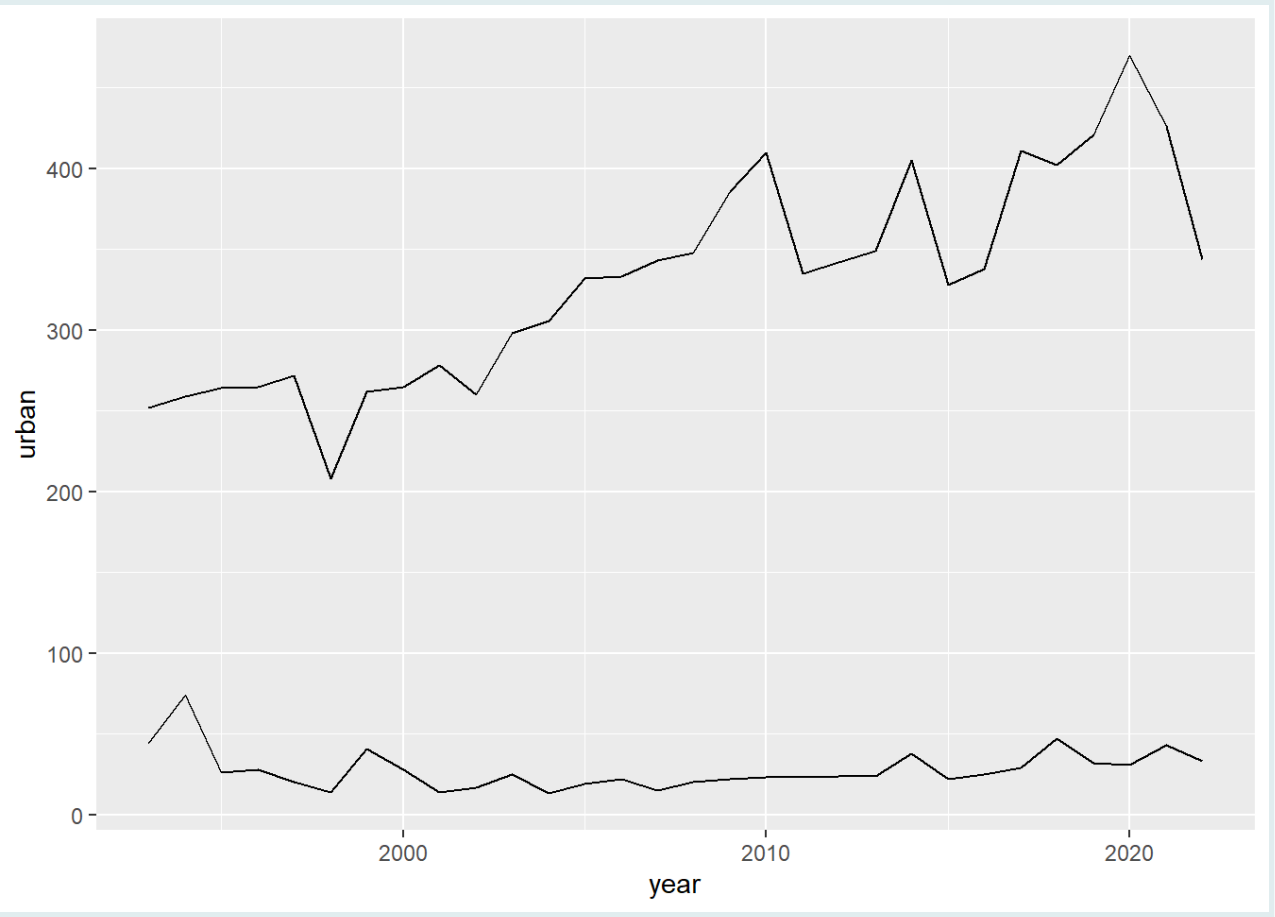
```r
annual_data_aus <- tb_data_aus %>%
  mutate(year = str_sub(period, 1, 4)) %>%
  mutate(year = as.numeric(year)) %>%
  # grouper par année
  group_by(year) %>%
  # sommer le nombre de cas chaque année
  summarise(rural = sum(rural),
            urban = sum(urban))
annual_data_aus
```

```
## # A tibble: 30 × 3
##     year rural urban
##    <dbl> <dbl> <dbl>
##  1  1993    44   252
##  2  1994    74   259
##  3  1995    26   264
##  4  1996    28   265
##  5  1997    20   272
##  6  1998    14   208
##  7  1999    41   262
##  8  2000    28   265
##  9  2001    14   278
```

```
## 10  2002    17   260
## # ℹ 20 more rows
```

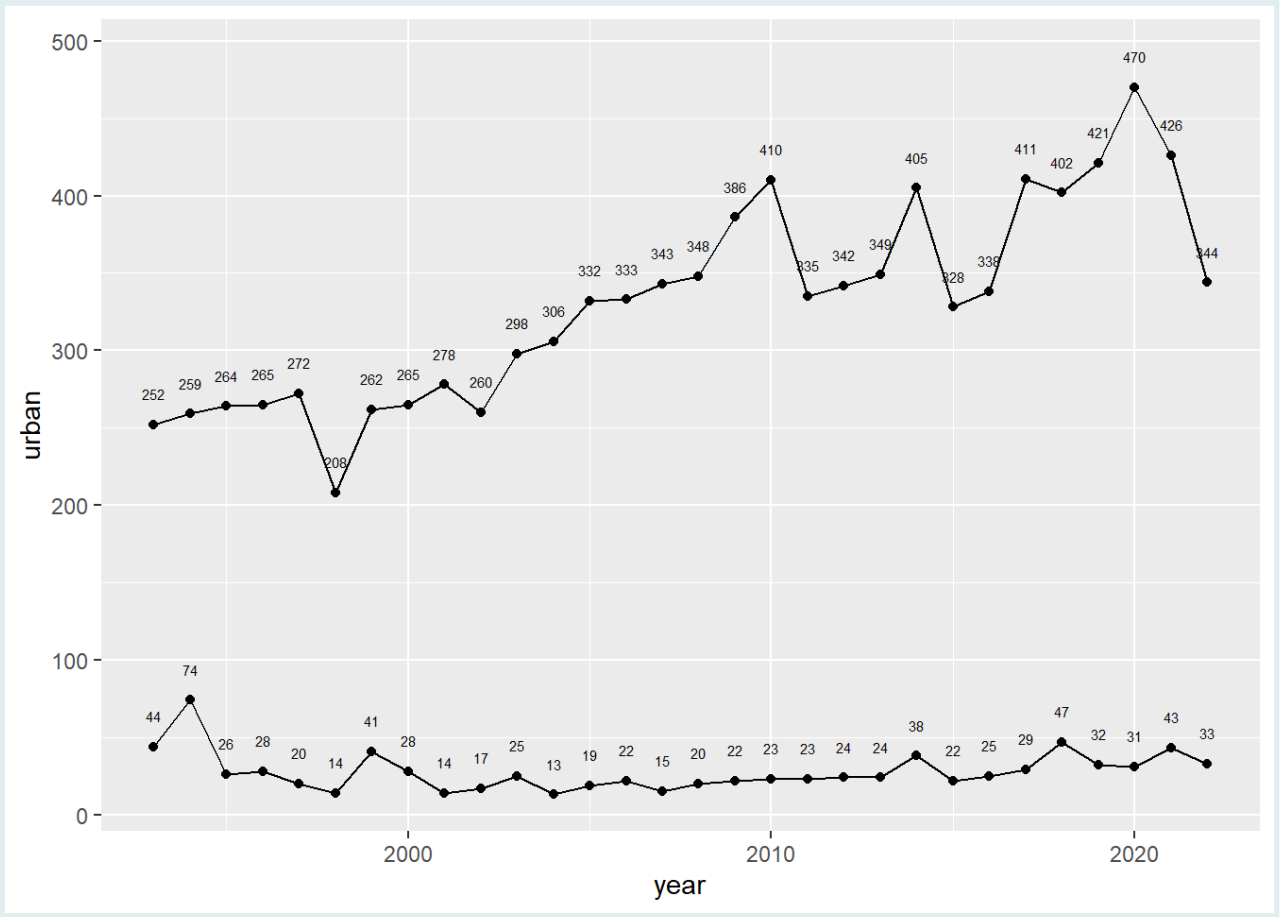
Maintenant que nous semblons avoir les données dans le format souhaité, faisons un premier graphique linéaire :

```
ggplot(annual_data_aus, aes(x = year)) +
  geom_line(aes(y = urban, couleur = "Urbain")) +
  geom_line(aes(y = rural, couleur = "Rural"))
```



C'est un graphique informatif, cependant, il y a une certaine redondance inutile dans le code, bien que vous ne le réalisiez peut-être pas encore. Cela deviendra plus clair si nous essayons d'ajouter d'autres géométries, comme des points ou du texte :

```
ggplot(annual_data_aus, aes(x = year)) +
  geom_line(aes(y = urban, couleur = "Urbain")) +
  geom_line(aes(y = rural, couleur = "Rural")) +
  geom_point(aes(y = urban, couleur = "Urbain")) +
  geom_point(aes(y = rural, couleur = "Rural")) +
  geom_text(aes(y = urban, label = urban), size = 2, nudge_y = 20) +
  geom_text(aes(y = rural, label = rural), size = 2, nudge_y = 20)
```



Comme vous pouvez le voir, nous devons répéter les mêmes lignes de code pour chaque géométrie. Cela est non seulement fastidieux, mais rend également le code plus difficile à lire et à interpréter. Si nous avions plus de deux catégories, comme cela arrive souvent, cela serait encore plus encombrant.

Heureusement, il existe une meilleure façon de faire. Nous pouvons utiliser la fonction `pivot_longer()` du package {tidyr} pour remodeler les données dans un format plus adapté pour la représentation graphique :

```
# Utilisation de `pivot_longer` de tidyr pour remodeler les données
annual_data_aus %>%
  pivot_longer(cols = c("urban", "rural"))
```

```
## # A tibble: 60 × 3
##     year name  value
##    <dbl> <chr> <dbl>
##  1  1993 urban   252
##  2  1993 rural    44
##  3  1994 urban   259
##  4  1994 rural    74
##  5  1995 urban   264
##  6  1995 rural    26
##  7  1996 urban   265
##  8  1996 rural    28
```

```
##  9  1997 urban   272
## 10  1997 rural    20
## # ℹ 50 more rows
```

Le code ci-dessus a converti les données d'un format "large" à un format "long". C'est un format plus adapté pour la représentation graphique, car il nous permet d'associer une colonne spécifique à l'esthétique `couleur`.

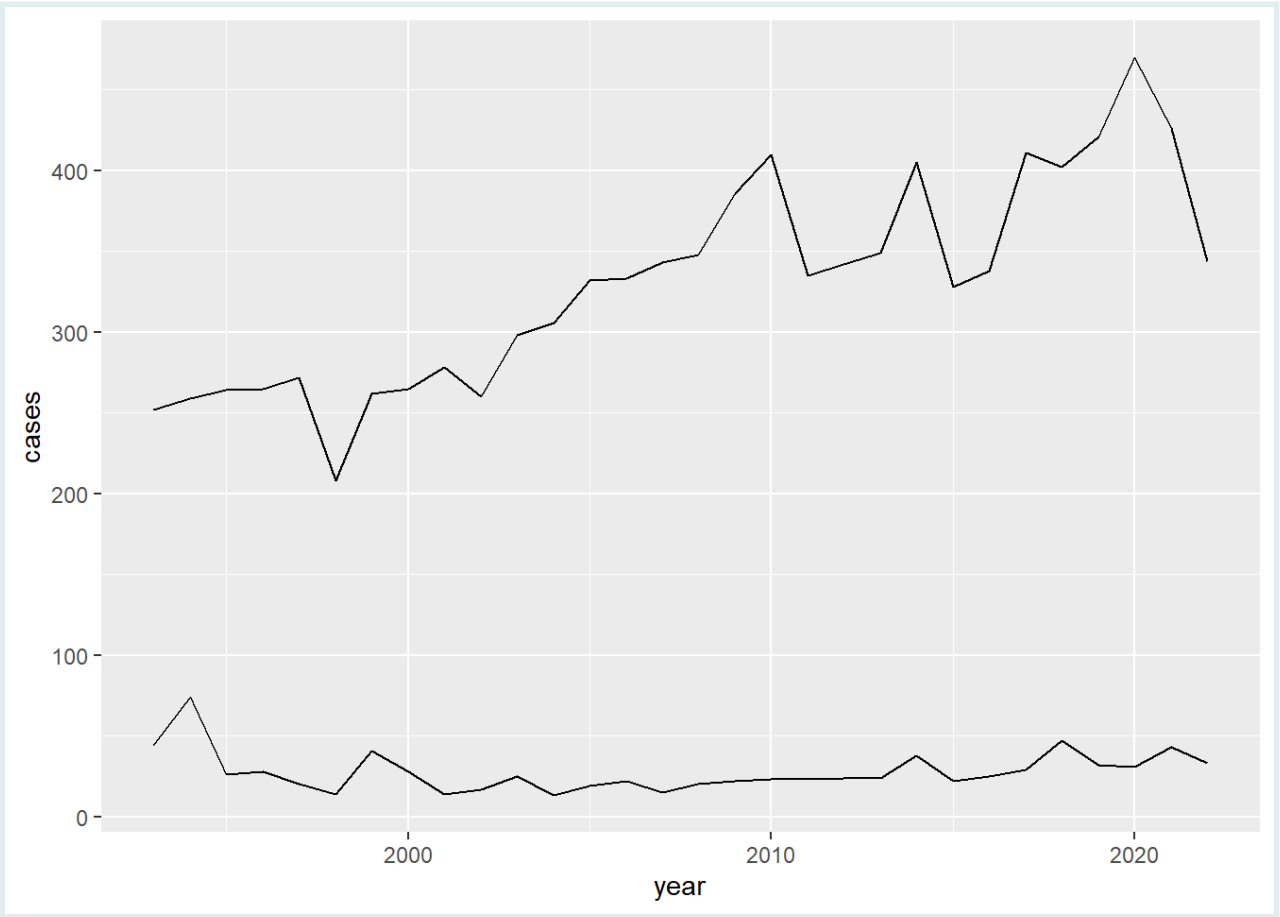
Avant de représenter ce jeu de données long, renommons les colonnes pour les rendre plus informatives :

```
aus_long <- annual_data_aus %>%
  pivot_longer(cols = c("urban", "rural")) %>%
  rename(region = name, cases = value)
```

Un Graphique Linéaire Groupé Basique

Nous sommes prêts à représenter à nouveau les données. Nous associons les esthétiques couleur et groupe à la colonne `region`, qui contient les deux catégories d'intérêt : urbain et rural.

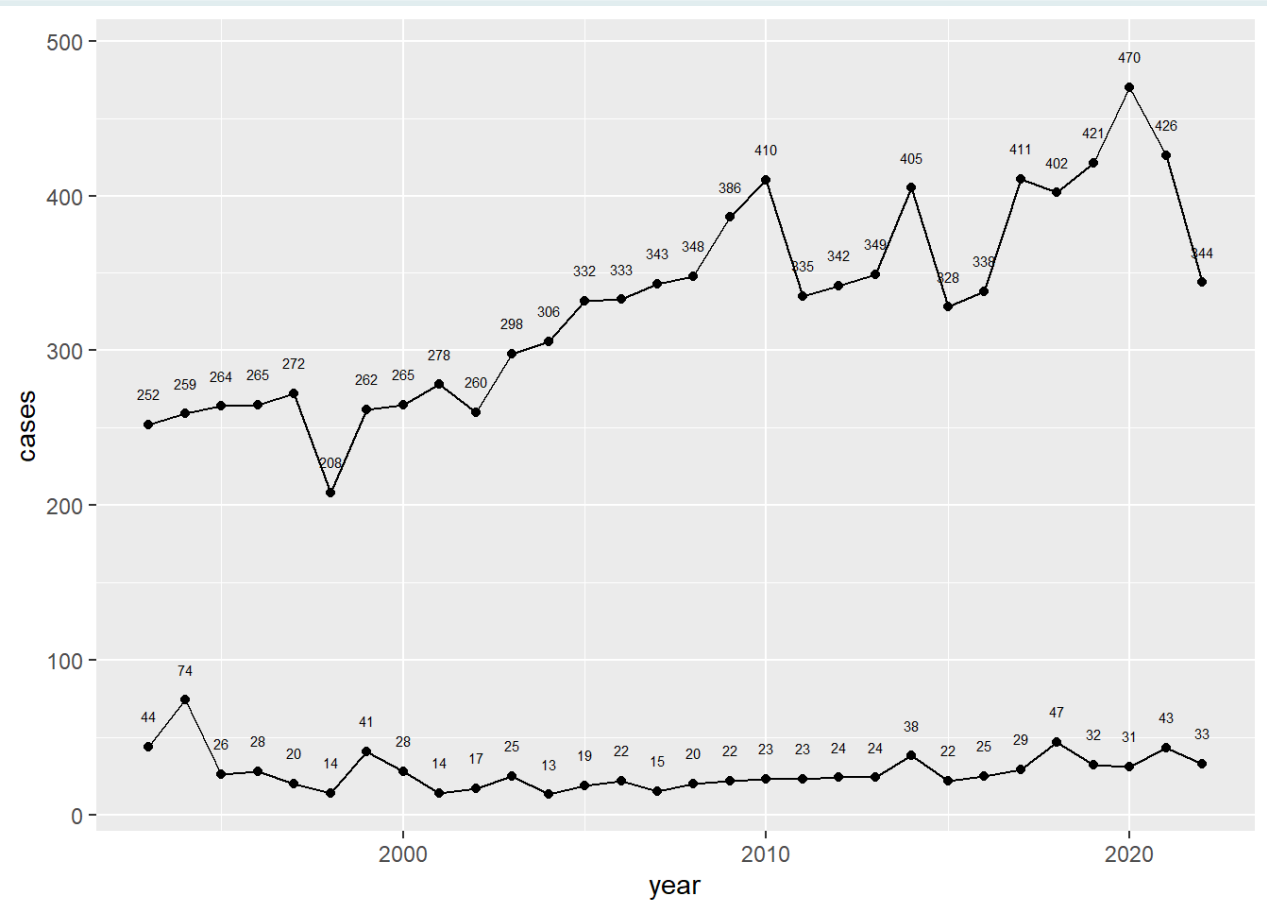
```
ggplot(aus_long, aes(x = year, y = cases, couleur = region, groupe = region))
        +
  geom_line()
```



Le code de tracé est maintenant plus concis, grâce à l'opération de pivotement effectuée précédemment.

Nous pouvons maintenant également ajouter des points et des étiquettes textuelles avec beaucoup moins de code :

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, group = region)) +
  geom_line() +
  geom_point() +
  geom_text(aes(label = cases), size = 2, nudge_y = 20)
```



Super ! Nous avons maintenant une vue claire des tendances des notifications annuelles de cas de tuberculose (TB) dans les zones rurales et urbaines au fil du temps. Cependant, il y a encore des améliorations esthétiques que nous pouvons apporter ; nous les aborderons dans la section suivante.

Q : Remodelage et Représentation des Données sur la Tuberculose

Considérez le jeu de données du Bénin montré ci-dessous, qui contient des informations sur les cas de tuberculose confirmés

bactériologiquement et diagnostiqués cliniquement pendant plusieurs années au Bénin. (Les données proviennent d'un article ici

```
tb_data_benin <- read_csv(here("data/benin_tb_notifs.csv"))
tb_data_benin
```

```
## # A tibble: 15 × 3
##     year new_clindx new_labconf
##    <dbl>      <dbl>       <dbl>
##  1  2000        289        2280
##  2  2001        286        2289
##  3  2002        338        2428
##  4  2003        350        2449
##  5  2004        330        2577
##  6  2005        346        2731
##  7  2006        261        2950
##  8  2007        294        2755
##  9  2008        239        2983
## 10  2009        279        2950
## 11  2010        307        2958
## 12  2011        346        3326
## 13  2012        277        3086
## 14  2013        285        3219
## 15  2014        318        3062
```

Remodelez le jeu de données en utilisant `pivot_longer()`, puis créez un graphique avec deux lignes, une pour chaque type de diagnostic de cas de tuberculose. Ajoutez des points et des étiquettes textuelles au graphique.

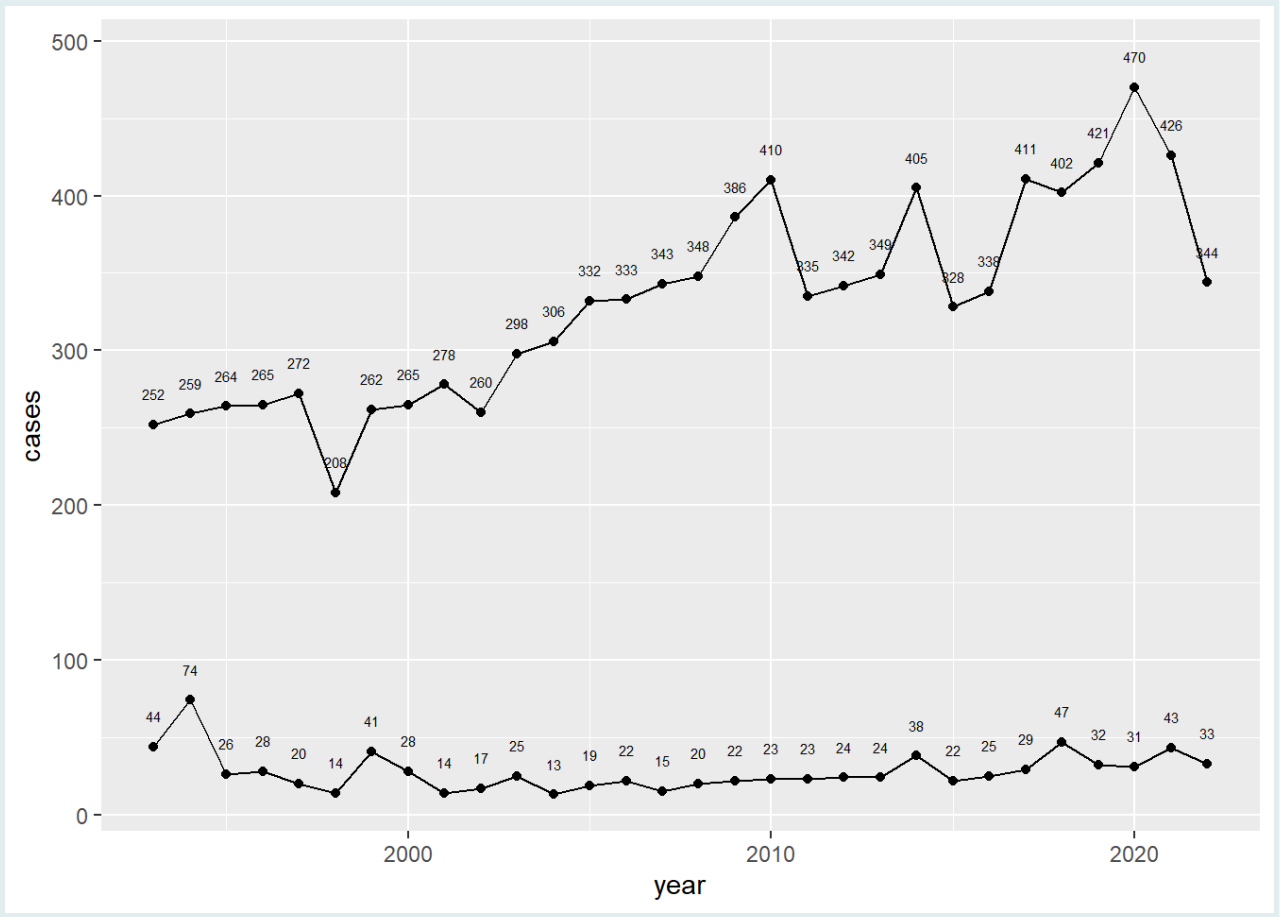
Améliorations Esthétiques des Graphiques Linéaires

Dans cette section, nous allons nous concentrer sur l'amélioration de l'esthétique des graphiques linéaires de séries temporelles pour renforcer leur clarté et leur attrait visuel.

Réduction de la Fréquence des Étiquettes

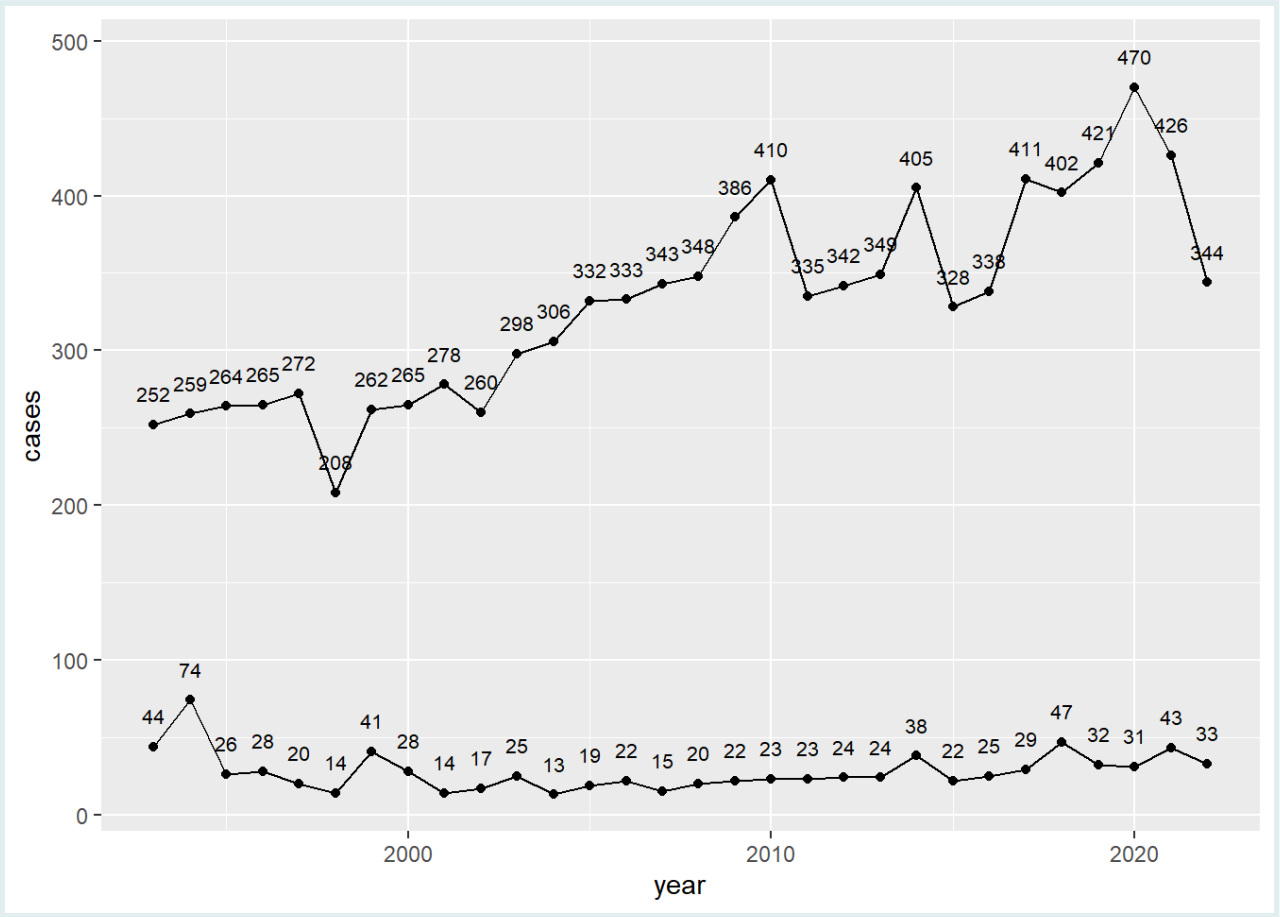
Lorsque nous avons laissé le graphique, il ressemblait à ceci :

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, groupe = region))
        +
  geom_line() +
  geom_point() +
  geom_text(aes(label = cases), size = 2, nudge_y = 20)
```



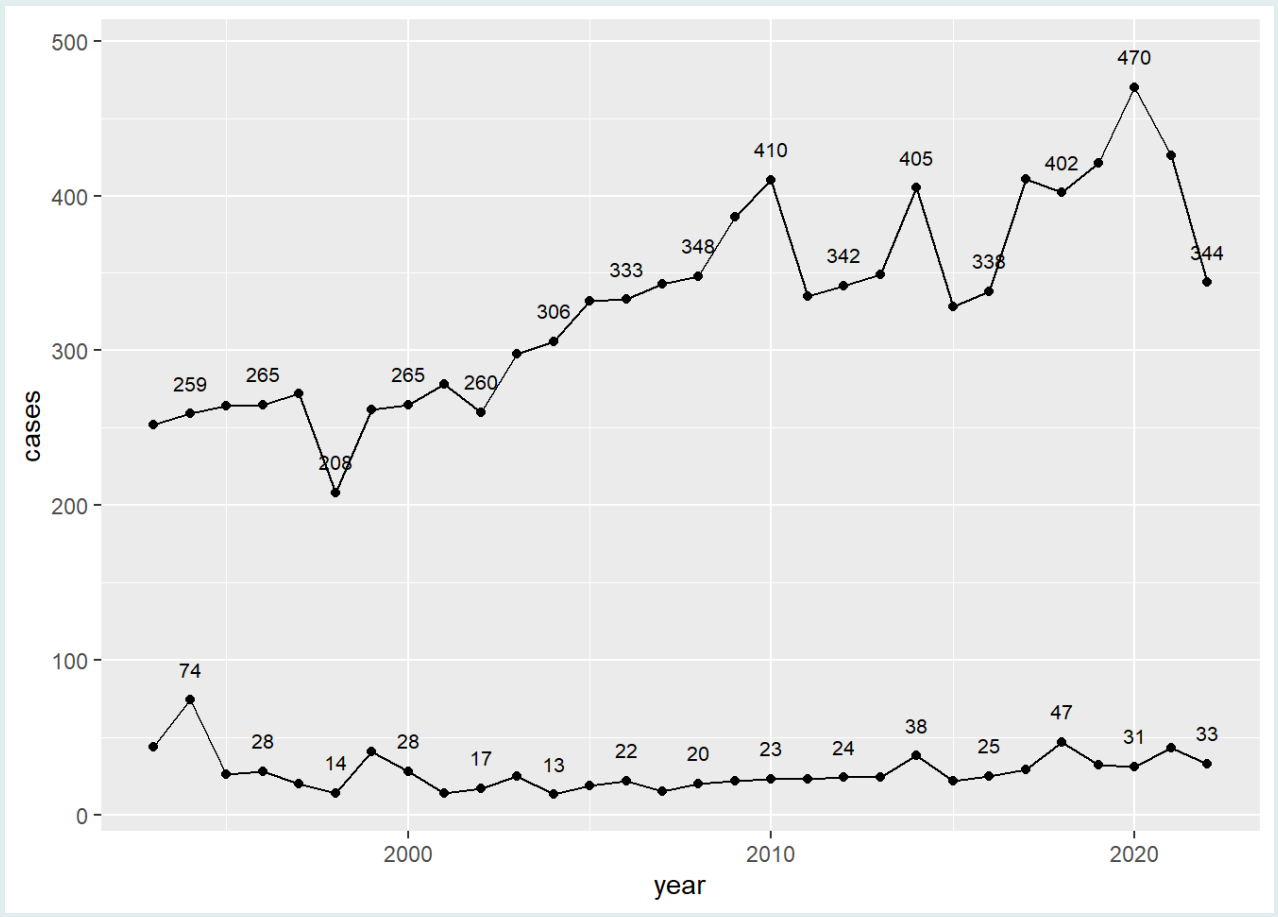
Un problème avec ce graphique est que les étiquettes textuelles sont un peu trop petites. De telles étiquettes minuscules ne sont pas idéales pour un graphique destiné au public, car elles sont difficiles à lire. Cependant, si nous augmentons la taille des étiquettes, celles-ci commenceront à se chevaucher, comme illustré ci-dessous :

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, groupe = region))
        +
  geom_line() +
  geom_point() +
  geom_text(aes(label = cases), size = 2.8, nudge_y = 20)
```



Pour éviter cet encombrement, une technique pratique consiste à afficher des étiquettes seulement pour certaines années. Pour ce faire, nous pouvons fournir un jeu de données personnalisé à la fonction `geom_text()`. Dans ce cas, nous allons créer un jeu de données qui contient uniquement les années paires :

```
even_years <- aus_long %>%
  filter(year %% 2 == 0) # Garder uniquement les années qui sont des multiples
        de 2

ggplot(aus_long, aes(x = year, y = cases, coulour = region, group = region)) +
  geom_line() +
  geom_point() +
  geom_text(data = even_years, aes(label = cases),
            size = 2.8, nudge_y = 20)
```



Super, nous avons maintenant des étiquettes plus grandes et elles ne se chevauchent pas.

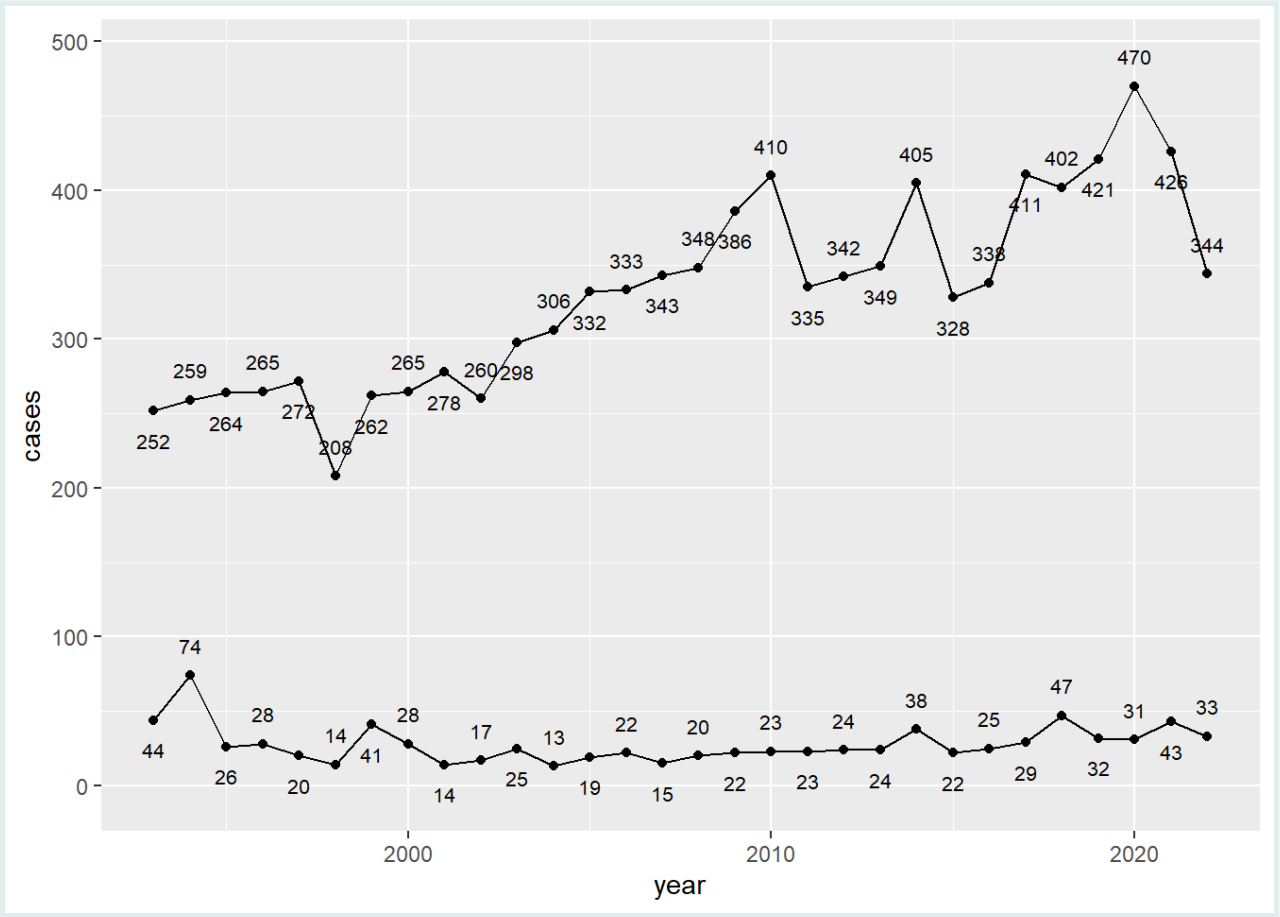
Alternance des Étiquettes

Bien que le graphique ci-dessus soit une amélioration, il serait encore mieux si nous pouvions afficher les étiquettes pour *toutes* les années. Nous pouvons le faire en affichant les étiquettes des années paires au-dessus des points de données, et les étiquettes des années impaires en dessous.

Inclure de nombreux points de données (dans la limite du raisonnable) dans vos graphiques est utile pour les responsables de la santé publique ; car ils peuvent rapidement tirer des chiffres du graphique lorsqu'ils essaient de prendre des décisions, sans avoir besoin de consulter les ensembles de données de référence.

Pour cela, créons un jeu de données filtré pour les années impaires, puis utilisons `geom_text()` deux fois, une fois pour chaque jeu de données filtré.

```
odd_years <- aus_long %>%
  filter(year %% 2 != 0) # Garder uniquement les années qui NE SONT PAS des
        multiples de 2

ggplot(aus_long, aes(x = year, y = cases, coulour = region, groupe = region))
        +
  geom_line() +
  geom_point() +
  geom_text(data = even_years, aes(label = cases),
            nudge_y = 20, size = 2.8) +
  geom_text(data = odd_years, aes(label = cases),
            nudge_y = -20, size = 2.8)
```



`ggrepel::geom_text_repel()`

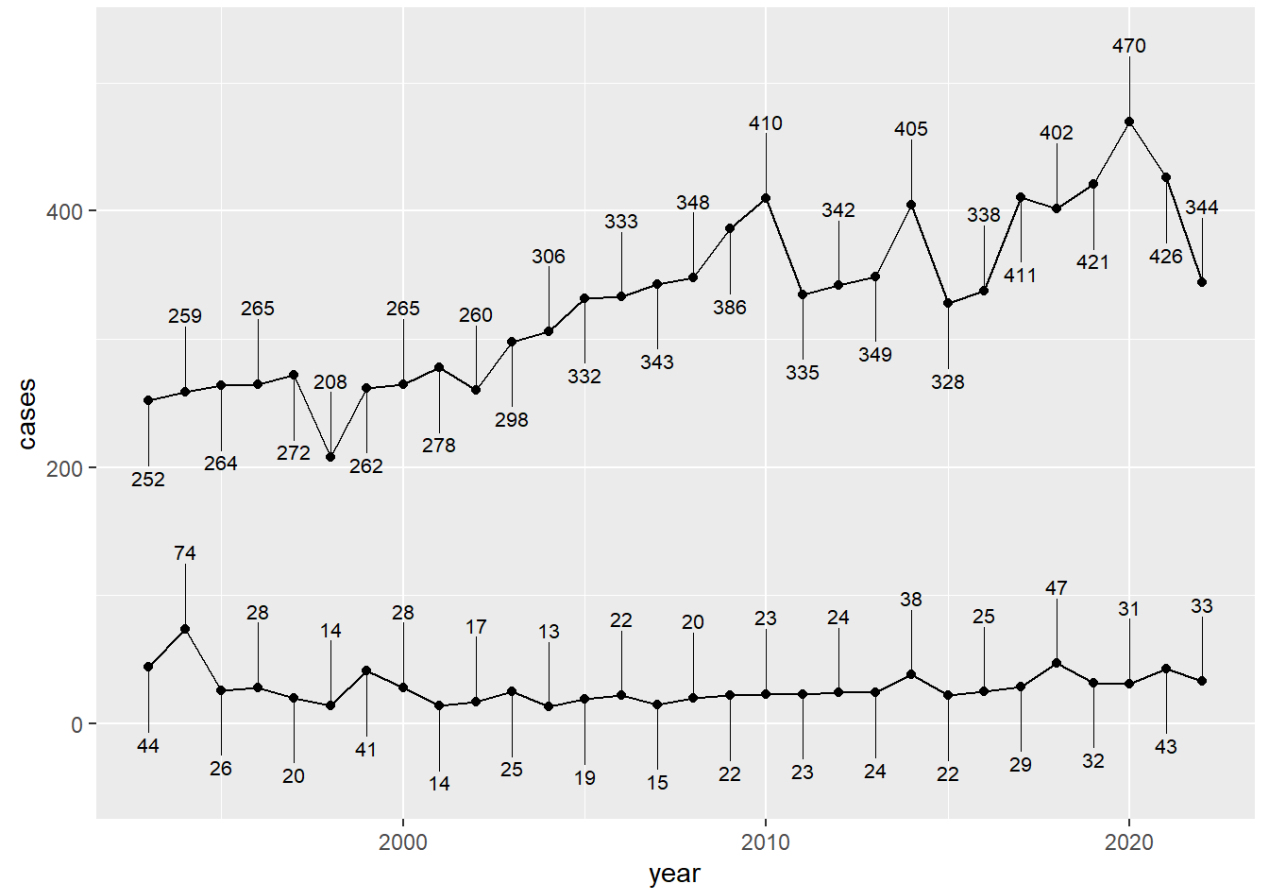
Le graphique ci-dessus est clair, mais il y a encore un certain chevauchement entre les étiquettes et la ligne.

Pour améliorer encore plus la clarté, nous pouvons utiliser la fonction `geom_text_repel()` du package {ggrepel}.

Cette fonction décale individuellement les étiquettes pour éviter le chevauchement et relie les étiquettes à leurs points de données avec des lignes, rendant plus facile de

voir à quel point de données chaque étiquette correspond, et nous permettant d'augmenter la distance entre les étiquettes et les points de données.

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, group = region)) +
  geom_line() +
  geom_point() +
  geom_text_repel(data = even_years, aes(label = cases),
                  nudge_y = 60, size = 2.8, segment.size = 0.1) +
  geom_text_repel(data = odd_years, aes(label = cases),
                  nudge_y = -60, size = 2.8, segment.size = 0.1)
```



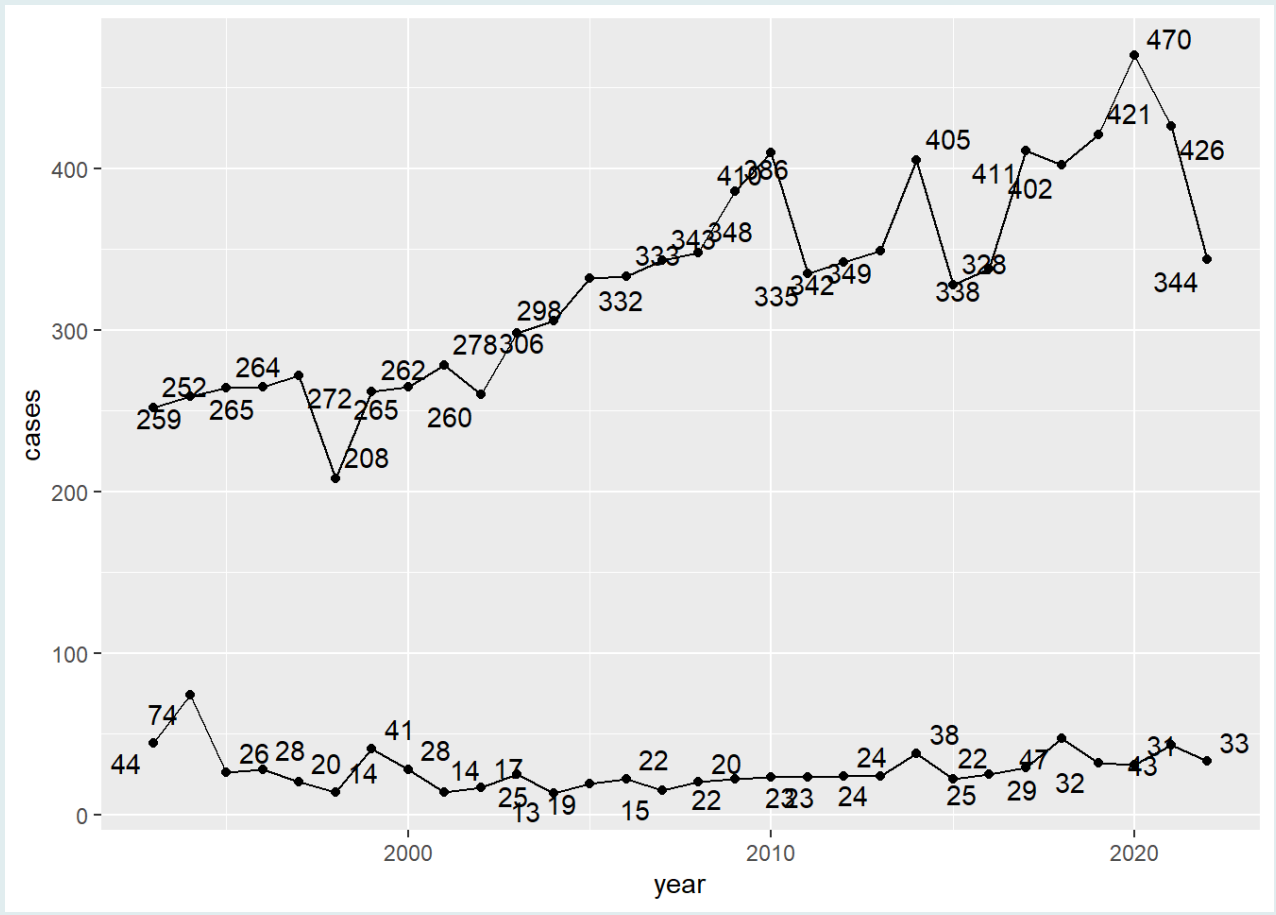
Comme vous pouvez le voir, la fonction `geom_text_repel()` prend essentiellement les mêmes arguments que `geom_text()`. L'argument supplémentaire, `segment.size`, contrôle la largeur des lignes reliant les étiquettes aux points de données.

Personnalisation de la Palette de Couleurs

Il est souvent utile de personnaliser la palette de couleurs de vos graphiques, pour qu'elle corresponde, par exemple, à la charte graphique de votre organisation.

Nous pouvons personnaliser les couleurs des lignes en utilisant la fonction `scale_color_manual()`. Ci-dessous, nous spécifions deux couleurs, une pour chaque région :

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, group = region)) +
  geom_line() +
  geom_point() +
  geom_text_repel(data = even_years, aes(label = cases),
                  décalage_y = 60, taille = 2.8, segment.size = 0.1) +
  geom_text_repel(data = odd_years, aes(label = cases),
                  décalage_y = -60, taille = 2.8, segment.size = 0.1) +
  scale_color_manual(values = c("urbain" = "#0fa3b1",
                                "rural" = "#2F2C4E"))
```

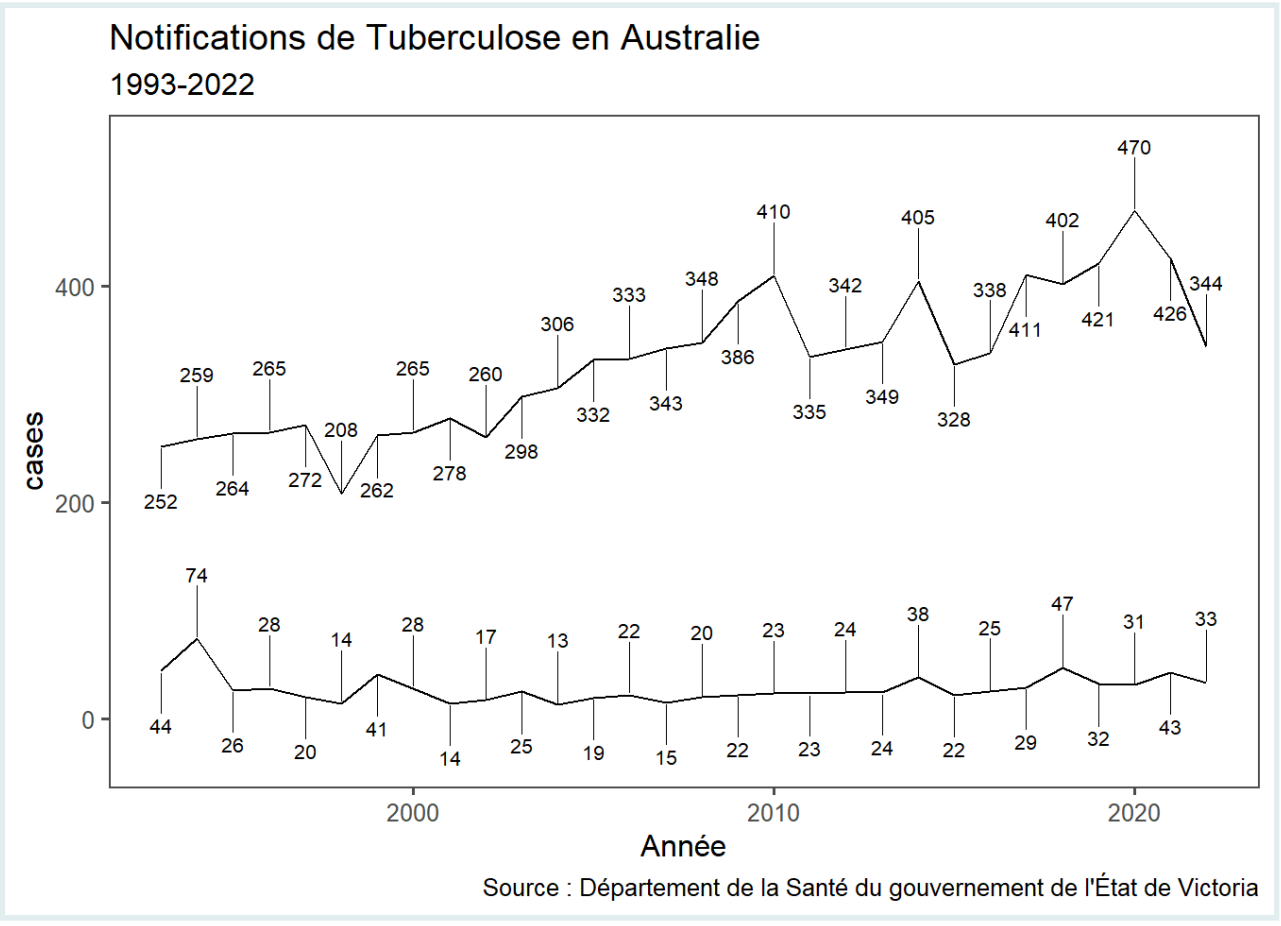


Succès !

Ajout d'Annotations au Graphique

Enfin, ajoutons une série de touches finales. Nous allons annoter le graphique avec des titres appropriés, des étiquettes d'axe et des légendes, et modifier le thème :

```
ggplot(aus_long, aes(x = year, y = cases, coulour = region, groupe = region))
        +
  geom_line(largeur_ligne = 1) +
  geom_text_repel(data = even_years, aes(label = cases),
                  nudge_y = 60, size = 2.8, segment.size = 0.08) +
  geom_text_repel(data = odd_years, aes(label = cases),
                  nudge_y = -50, size = 2.8, segment.size = 0.08) +
  scale_color_manual(values = c("urbain" = "#0fa3b1", "rural" = "#2F2C4E")) +
  labs(title = "Notifications de Tuberculose en Australie",
       subtitle = "1993-2022",
       caption = "Source : Département de la Santé du gouvernement de l'État
        de Victoria",
       x = "Année",
       coulour = "Région") +
  ggthemes::theme_few() +
  theme(legend.position = "droite")
```



Cela couvre certaines options pour améliorer l'esthétique des graphiques linéaires ! N'hésitez pas à ajuster davantage les visuels en fonction de vos besoins d'analyse spécifiques.

RECAP

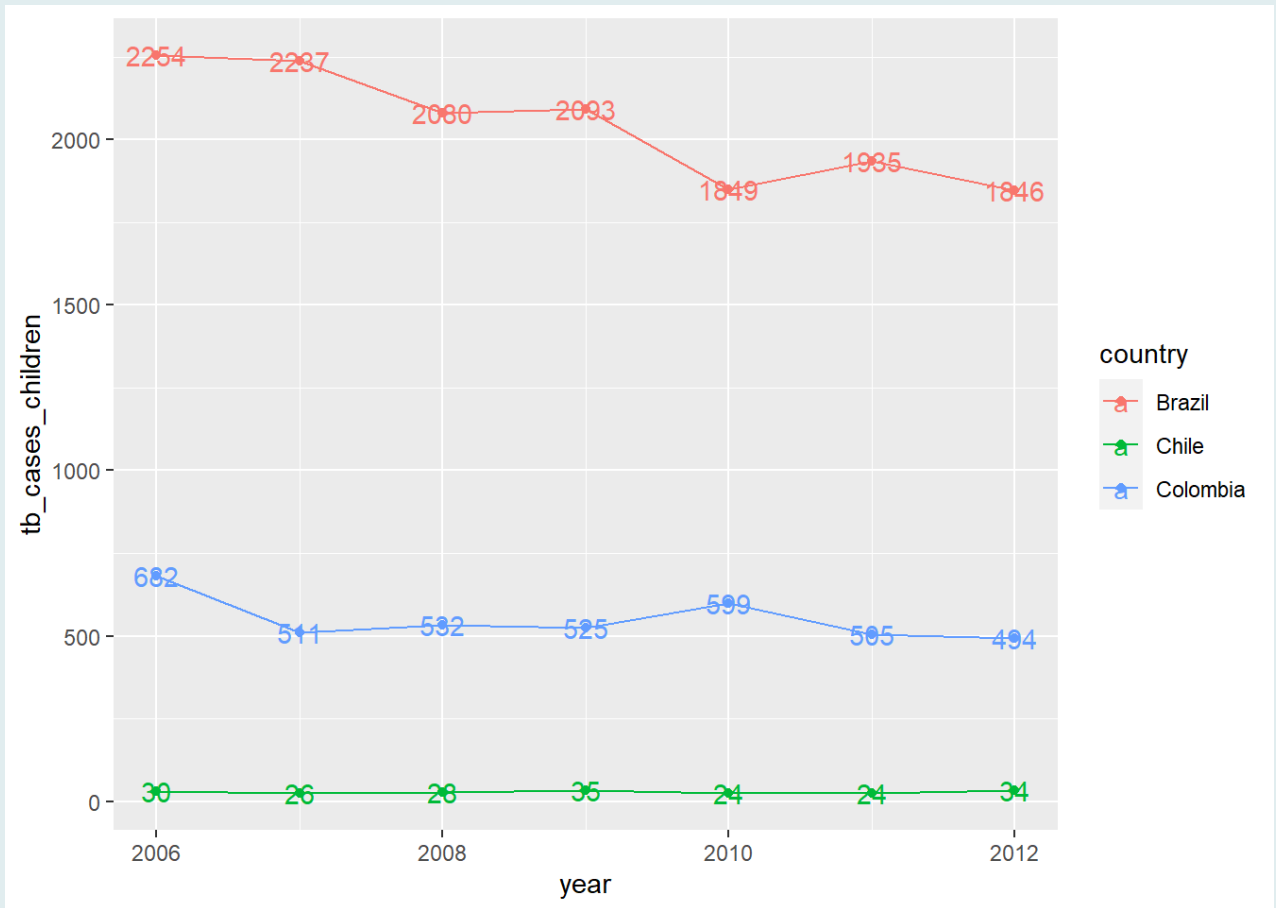
RECAP

de tuberculose en Australie. Nous avons équilibré le besoin d'informations détaillées avec une présentation claire, rendant notre graphique à la fois informatif et accessible.

Q : Améliorations esthétiques

Considérez le graphique suivant, qui montre le nombre de cas de tuberculose chez les enfants dans trois pays au fil du temps :

```
tb_child_cases_southam <- tidyr::who2 %>%
  transmute(country, year,
            tb_cases_children = sp_m_014 + sp_f_014 + sn_m_014 + sn_f_014) %>%
  filter(country %in% c("Brazil", "Colombia", "Chile")) %>%
  filter(!is.na(tb_cases_children))

tb_child_cases_southam %>%
  ggplot(aes(x = year, y = tb_cases_children, color = country)) +
  geom_line() +
  geom_point() +
  geom_text(aes(label = tb_cases_children))
```



Améliorez ce graphique en implémentant les améliorations suivantes :

- Réglez les étiquettes `geom_text` pour alterner au-dessus et en dessous des lignes, semblable à l'exemple que nous avons vu ci-dessus.
- Utilisez la palette de couleurs suivante `c("#212738", "#F97068", "#067BC2")`
- Appliquez `theme_classic()`
- Ajoutez un titre, un sous-titre et une légende pour fournir un contexte et des informations sur les données. (Vous pouvez taper `?tidyr::who2` dans la console pour en savoir plus sur la source des données.)

~~Représent~~ation des Intervalle de Confiance avec `geom_ribbon()`

Dans les visualisations de séries temporelles, il est souvent important de représenter les intervalles de confiance pour indiquer le niveau d'incertitude de vos données.

Nous allons démontrer comment faire cela en utilisant un jeu de données sur les nouvelles infections par le VIH au Brésil, qui comprend les nombres estimés pour les cas masculins et féminins ainsi que les intervalles de confiance. Le jeu de données est issu de l'Organisation Mondiale de la Santé (OMS) et peut être consulté ici.

Commençons par charger et inspecter le jeu de données :

```
hiv_data_brazil <-
  rio::import(here("data/new_hiv_infections_gho.xlsx"),
                                  sheet = "Brazil") %>%
  as_tibble() %>%
  janitor::clean_names()
hiv_data_brazil
```

```
## # A tibble: 89 × 5
##    continent country  year sex
##    <chr>     <chr>   <dbl> <chr>
##  1 Americas  Brazil   2022 Female
##  2 Americas  Brazil   2022 Male
##  3 Americas  Brazil   2022 Both sexes
##  4 Americas  Brazil   2021 Female
##  5 Americas  Brazil   2021 Male
##  6 Americas  Brazil   2021 Both sexes
##  7 Americas  Brazil   2020 Female
##  8 Americas  Brazil   2020 Male
##  9 Americas  Brazil   2020 Both sexes
## 10 Americas  Brazil   2019 Female
##    new_hiv_cases
##    <chr>
##  1 13 000 [12 000 - 15 000]
##  2 37 000 [35 000 - 40 000]
##  3 51 000 [47 000 - 54 000]
##  4 14 000 [12 000 - 15 000]
```

```
##  5 37 000 [34 000 - 39 000]
##  6 50 000 [47 000 - 53 000]
##  7 14 000 [13 000 - 15 000]
##  8 35 000 [32 000 - 37 000]
##  9 49 000 [46 000 - 52 000]
## 10 14 000 [13 000 - 15 000]
## # ℹ 79 more rows
```

Nous pouvons voir que la colonne `new_hiv_cases` contient à la fois le nombre de cas et les intervalles de confiance correspondants entre crochets. Ce format ne peut pas être utilisé directement pour la représentation graphique, donc nous devons les extraire sous forme numérique pure.

Tout d'abord, pour séparer ces valeurs, nous pouvons utiliser la fonction `separate()` du package {tidyr} :

```
hiv_data_brazil %>%
  separate(new_hiv_cases,
           into = c("cases", "cases_lower", "cases_upper"),
           sep = "\\[|-")
```

```
## # A tibble: 89 × 7
##    continent country  year sex        cases
##    <chr>     <chr>   <dbl> <chr>      <chr>
##  1 Americas  Brazil   2022 Female     "13 000 "
##  2 Americas  Brazil   2022 Male       "37 000 "
##  3 Americas  Brazil   2022 Both sexes "51 000 "
##  4 Americas  Brazil   2021 Female     "14 000 "
##  5 Americas  Brazil   2021 Male       "37 000 "
##  6 Americas  Brazil   2021 Both sexes "50 000 "
##  7 Americas  Brazil   2020 Female     "14 000 "
##  8 Americas  Brazil   2020 Male       "35 000 "
##  9 Americas  Brazil   2020 Both sexes "49 000 "
## 10 Americas  Brazil   2019 Female     "14 000 "
##    cases_lower cases_upper
##    <chr>       <chr>
##  1 "12 000 "   " 15 000]"
##  2 "35 000 "   " 40 000]"
##  3 "47 000 "   " 54 000]"
##  4 "12 000 "   " 15 000]"
##  5 "34 000 "   " 39 000]"
##  6 "47 000 "   " 53 000]"
##  7 "13 000 "   " 15 000]"
##  8 "32 000 "   " 37 000]"
##  9 "46 000 "   " 52 000]"
## 10 "13 000 "   " 15 000]"
## # ℹ 79 more rows
```

Dans le code ci-dessus, nous divisons la colonne `cases_upper` en trois nouvelles colonnes : `cases`, `cases_lower` et `cases_upper`. Nous utilisons `[` et `-` comme séparateurs. Le double antislash `\\` est utilisé pour échapper au crochet, qui a une

signification spéciale dans les expressions régulières. Et le `|` est utilisé pour indiquer que soit `[` soit `-` peut être utilisé comme séparateur.

Les modèles de langage de grande taille comme ChatGPT sont excellents pour comprendre les expressions régulières. Si vous êtes bloqué avec un code comme `sep = "\\[|-"` et que vous voulez comprendre ce qu'il fait, vous pouvez demander à ChatGPT de vous l'expliquer. Et si vous avez besoin de générer de telles expressions vous-même, vous pouvez demander à ChatGPT de les générer pour vous.

Ensuite, nous devons convertir ces valeurs en chaîne de caractères en valeurs numériques, en supprimant tous les caractères non numériques.

```
hiv_data_brazil_clean <-
  hiv_data_brazil %>%
  separate(new_hiv_cases,
           into = c("cases", "cases_lower", "cases_upper"),
           sep = "\\[|-") %>%
  mutate(across(c("cases", "cases_lower", "cases_upper"),
                ~ str_replace_all(.x, "[^0-9]", "") %>%
                  as.numeric()))

hiv_data_brazil_clean
```

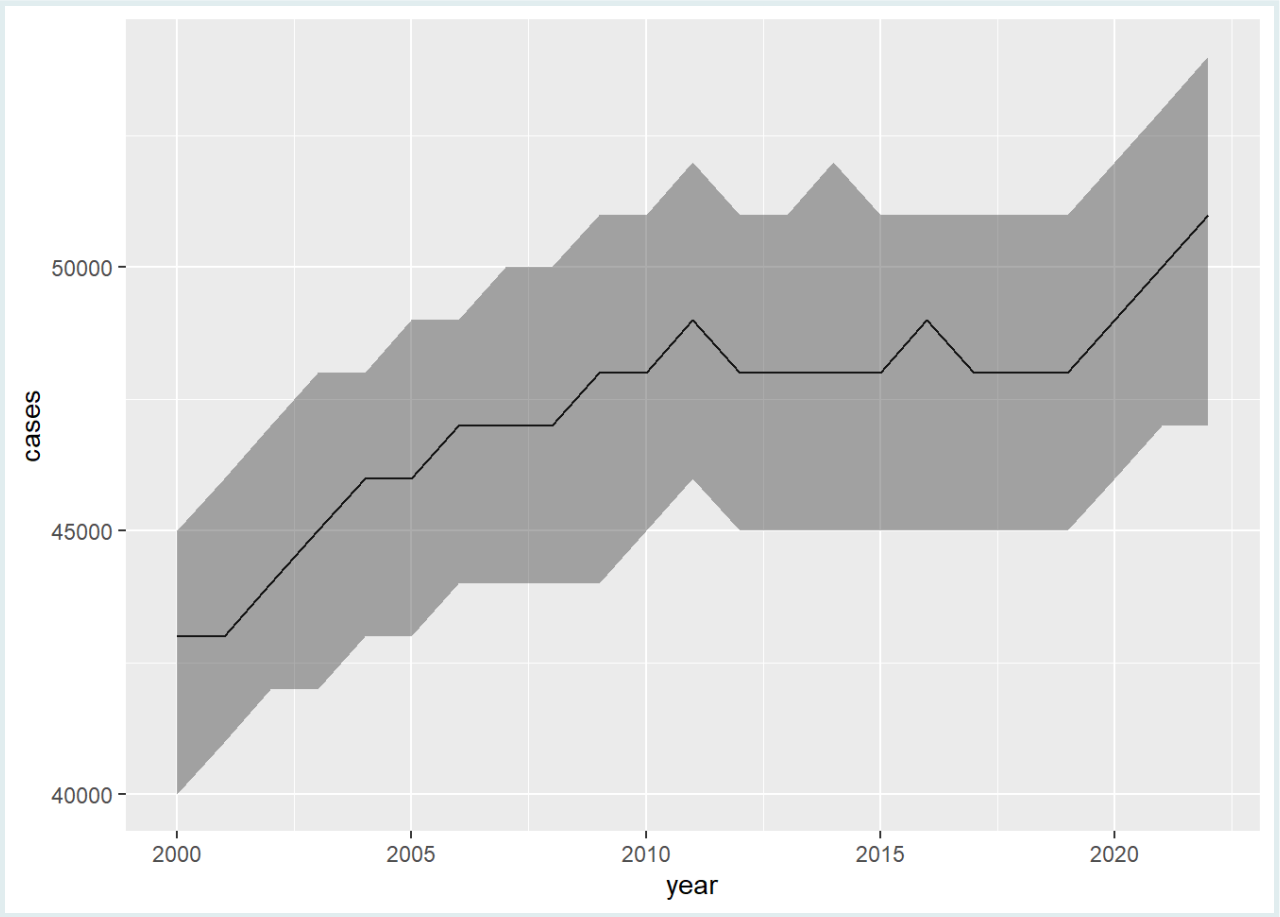
```
## # A tibble: 89 × 7
##    continent country  year sex        cases cases_lower
##    <chr>     <chr>   <dbl> <chr>      <dbl>       <dbl>
##  1 Americas  Brazil   2022 Female     13000       12000
##  2 Americas  Brazil   2022 Male       37000       35000
##  3 Americas  Brazil   2022 Both sexes 51000       47000
##  4 Americas  Brazil   2021 Female     14000       12000
##  5 Americas  Brazil   2021 Male       37000       34000
##  6 Americas  Brazil   2021 Both sexes 50000       47000
##  7 Americas  Brazil   2020 Female     14000       13000
##  8 Americas  Brazil   2020 Male       35000       32000
##  9 Americas  Brazil   2020 Both sexes 49000       46000
## 10 Americas  Brazil   2019 Female     14000       13000
##    cases_upper
##          <dbl>
##  1       15000
##  2       40000
##  3       54000
##  4       15000
##  5       39000
##  6       53000
##  7       15000
##  8       37000
##  9       52000
```

```
## 10       15000
## # ℹ 79 more rows
```

Le code ci-dessus semble complexe, mais essentiellement, il nettoie les données en ne conservant que les caractères numériques, puis convertit ces nombres en valeurs numériques réelles. Voir notre leçon sur la fonction `across()` pour plus de détails.

Nous sommes enfin prêts à représenter les données. Nous utiliserons `geom_ribbon()` de ggplot pour afficher les intervalles de confiance :

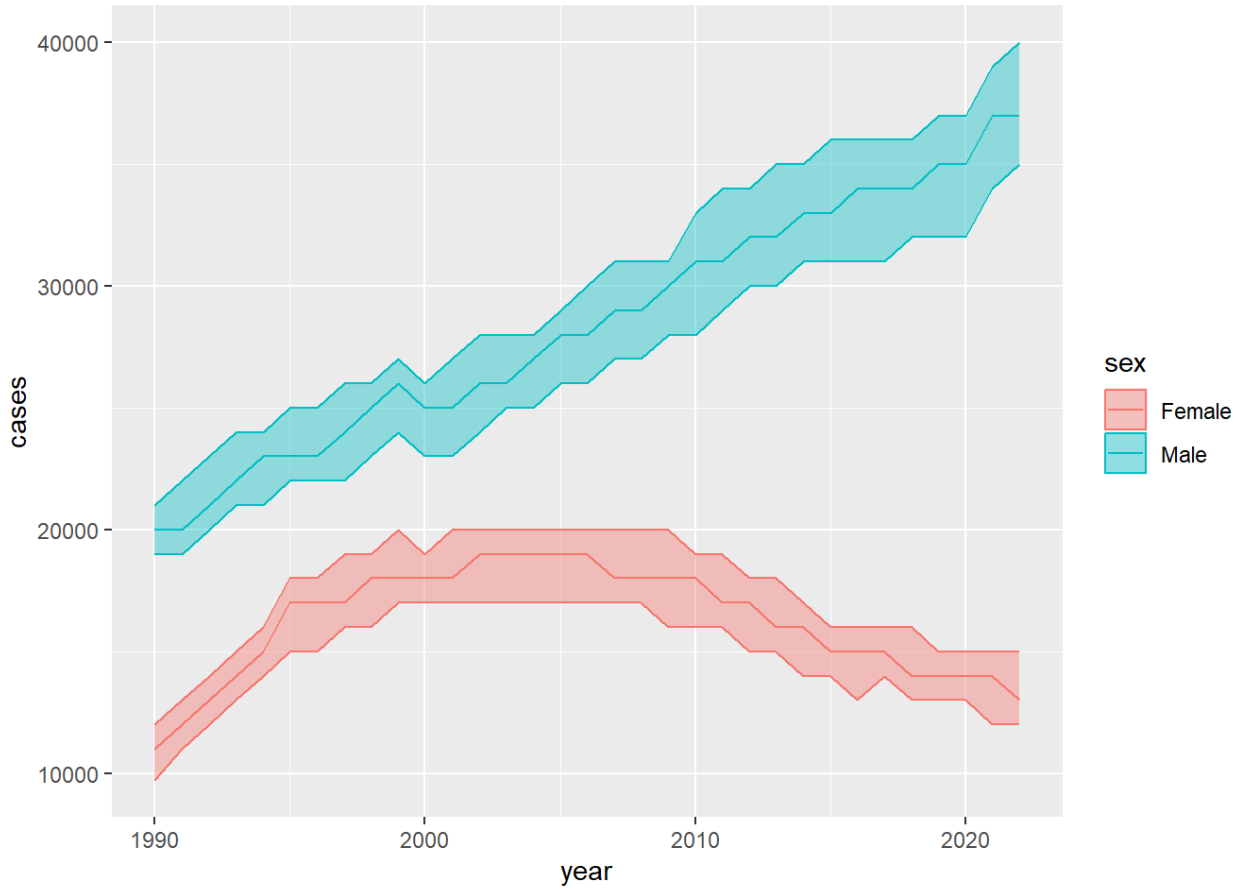
```
hiv_data_brazil_clean %>%
  filter(sex == "Both sexes") %>%
  ggplot(aes(x = year, y = cases)) +
  geom_line() +
  geom_ribbon(aes(ymin = cases_lower, ymax = cases_upper), alpha = 0.4)
```



La fonction `geom_ribbon()` prend les esthétiques `x` et `y` comme `geom_line()`, mais elle prend également en compte les esthétiques `ymin` et `ymax`, pour déterminer l'étendue verticale du ruban. Nous réglons également la transparence du ruban à l'aide de l'argument `alpha`.

Nous pouvons créer un ruban séparé pour les hommes et les femmes pour comparer leurs tendances d'infection.

```
hiv_data_brazil_clean %>%
  filter(sex != "Both sexes") %>%
  ggplot(aes(x = year, y = cases, color = sex, fill = sex)) +
  geom_line() +
  geom_ribbon(aes(ymin = cases_lower, ymax = cases_upper), alpha = 0.4)
```



Il est à noter que les taux d'infection par le VIH chez les femmes ont diminué ces dernières années, mais ceux chez les hommes ont augmenté.

Q : Représentation des intervalles de confiance

Considérez le jeu de données suivant qui montre le nombre de cas annuels de paludisme au Kenya et au Nigeria. Les données proviennent du dépôt de données de l'Observatoire mondial de la santé de l'OMS et peuvent être consultées ici.

```
nig_ken_mal <- read_csv("data/nigeria_kenya_malaria.csv")
nig_ken_mal
```

```
## # A tibble: 44 × 3
##    country  year malaria_cases
##    <chr>   <dbl> <chr>
##  1 Kenya    2021 3 419 698 (2 478 000 to 4 641 000)
##  2 Nigeria  2021 65 399 501 (47 520 000 to 89 000 000)
##  3 Kenya    2020 3 302 189 (2 385 000 to 4 466 000)
##  4 Nigeria  2020 65 133 759 (46 800 000 to 88 650 000)
##  5 Kenya    2019 3 037 541 (2 168 000 to 4 130 000)
##  6 Nigeria  2019 61 379 283 (47 440 000 to 78 810 000)
##  7 Kenya    2018 3 068 062 (2 148 000 to 4 260 000)
##  8 Nigeria  2018 59 652 248 (46 930 000 to 75 230 000)
##  9 Kenya    2017 3 155 636 (2 217 000 to 4 375 000)
## 10 Nigeria  2017 57 869 533 (45 870 000 to 72 050 000)
## # ℹ 34 more rows
```

Écrivez du code pour extraire les intervalles de confiance de la colonne “malaria_cases” et créez un graphique avec des intervalles de confiance en utilisant `geom_ribbon()`. Utilisez une couleur différente pour chaque pays.

Lissage des Tendances Bruyantes

Lors de l’analyse de données de séries temporelles, il est courant que les mesures quotidiennes ou granulaires montrent beaucoup de bruit et de variabilité, ce qui peut masquer les tendances importantes qui nous intéressent réellement. Les techniques de lissage peuvent aider à mettre en évidence ces tendances et motifs. Nous allons explorer plusieurs techniques à cet effet dans les sections ci-dessous.

Tout d’abord, faisons quelques préparations de données !

Création d’un Tableau d’Incidence à partir d’une Liste de Cas

Considérez la liste de cas suivante des admissions pédiatriques pour la malaria dans quatre hôpitaux au Mozambique (Source de données) :

```
mal <-
  rio::import(here("data/pediatric_malaria_data_joao_2021.xlsx")) %>%
  as_tibble() %>%
  mutate(date_positive_test = as.Date(date_positive_test)) %>%
  # Keep data from 2019-2020
  filter(date_positive_test >= as.Date("2019-01-01"),
         date_positive_test <= as.Date("2020-12-31"))
mal
```

```
## # A tibble: 20,939 × 4
##    date_positive_test neighbourhood sex   age
##    <date>             <chr>         <chr> <chr>
##  1 2020-01-22         25 de Junho 1 M     6-11 meses
##  2 2020-01-22         Chicueu       F     5-14 anos
##  3 2020-01-22         Mussessa      M     5-14 anos
##  4 2020-01-22         Nhamizara     M     12-23 meses
##  5 2020-01-22         Nhamizara     F     12-23 meses
##  6 2020-01-22         Unidade       F     5-14 anos
##  7 2020-01-22         Bapua         F     12-23 meses
##  8 2020-01-22         Bapua         F     5-14 anos
##  9 2020-01-22         7 de Abril    M     12-23 meses
## 10 2020-01-22         Nhanguzue     F     5-14 anos
## # ℹ 20,929 more rows
```

Chaque ligne correspond à un cas unique de malaria, et la colonne `date_test_positif` indique la date à laquelle l'enfant a été testé positif pour la malaria.

Pour obtenir un décompte des cas par jour - c'est-à-dire, un tableau d'incidence - nous pouvons simplement utiliser `count()` pour agréger les cas par date de test positif :

```
mal %>%
  count(date_positive_test, name = "cases")
```

```
## # A tibble: 235 × 2
##    date_positive_test cases
##    <date>             <int>
##  1 2019-01-01            67
##  2 2019-01-02           120
##  3 2019-01-03           112
##  4 2019-01-04           203
##  5 2019-01-05            85
##  6 2019-01-07           115
##  7 2019-01-08           196
##  8 2019-01-10            89
##  9 2019-01-11            55
## 10 2019-01-12            69
## # ℹ 225 more rows
```

Cependant, de nombreuses dates manquent - les jours où aucun enfant n'a été admis. Pour créer un tableau d'incidence complet, nous devrions utiliser `complete()` pour insérer les dates manquantes, puis remplir les valeurs manquantes avec 0 :

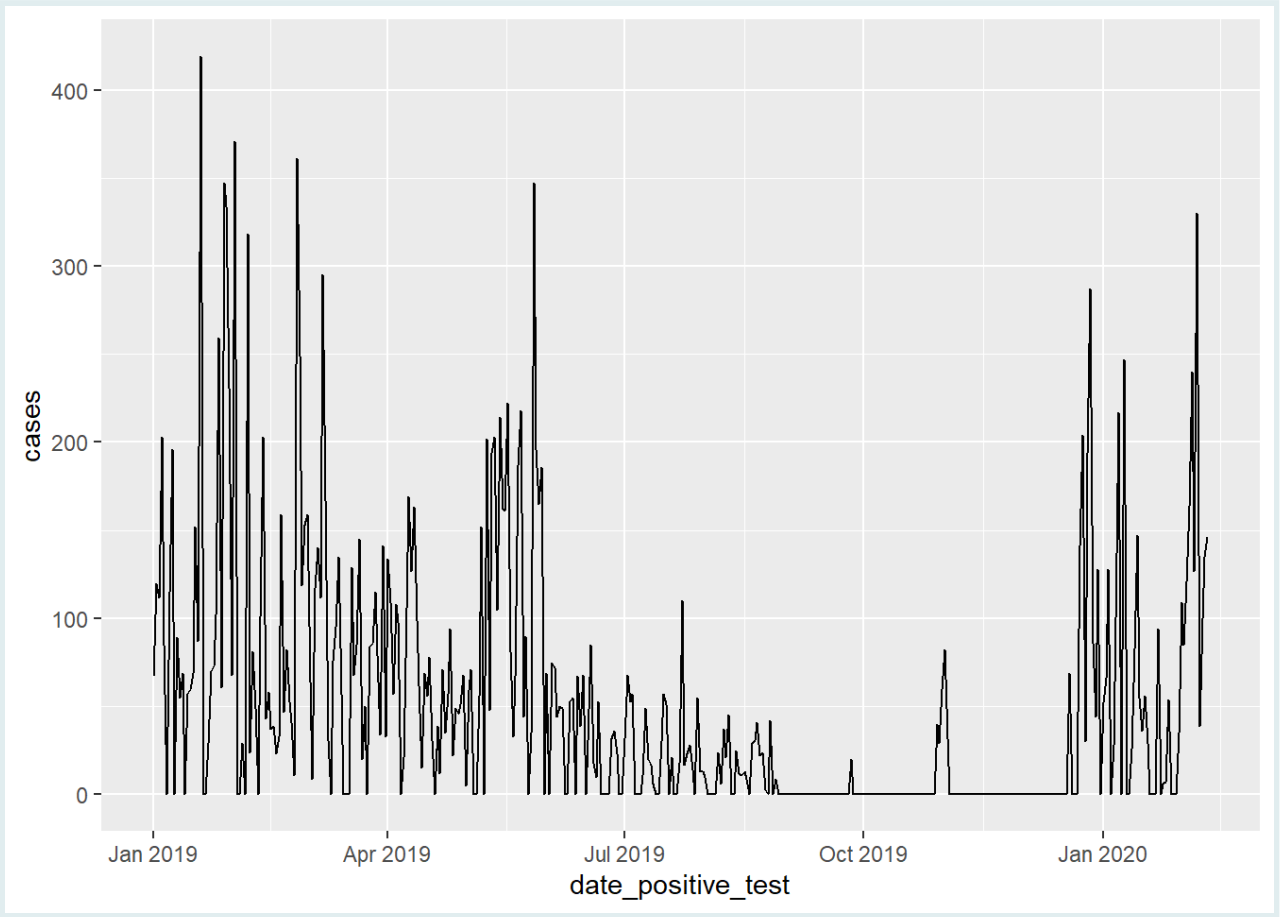
```
mal_notif_count <- mal %>%
  count(date_positive_test, name = "cases") %>%
  complete(date_positive_test = seq.Date(min(date_positive_test),
                                         max(date_positive_test),
                                         by = "day"),
           fill = list(cases = 0))

mal_notif_count
```

```
## # A tibble: 406 × 2
##    date_positive_test cases
##    <date>             <int>
##  1 2019-01-01            67
##  2 2019-01-02           120
##  3 2019-01-03           112
##  4 2019-01-04           203
##  5 2019-01-05            85
##  6 2019-01-06             0
##  7 2019-01-07           115
##  8 2019-01-08           196
##  9 2019-01-09             0
## 10 2019-01-10            89
## # ℹ 396 more rows
```

Maintenant, nous avons un tableau d'incidence complet avec le nombre de cas sur 406 jours consécutifs.

Nous pouvons maintenant tracer les données pour voir la tendance globale :

```
# Créer une épicourbe basique en utilisant ggplot2
ggplot(mal_notif_count, aes(x = date_positive_test, y = cases)) +
  geom_line()
```

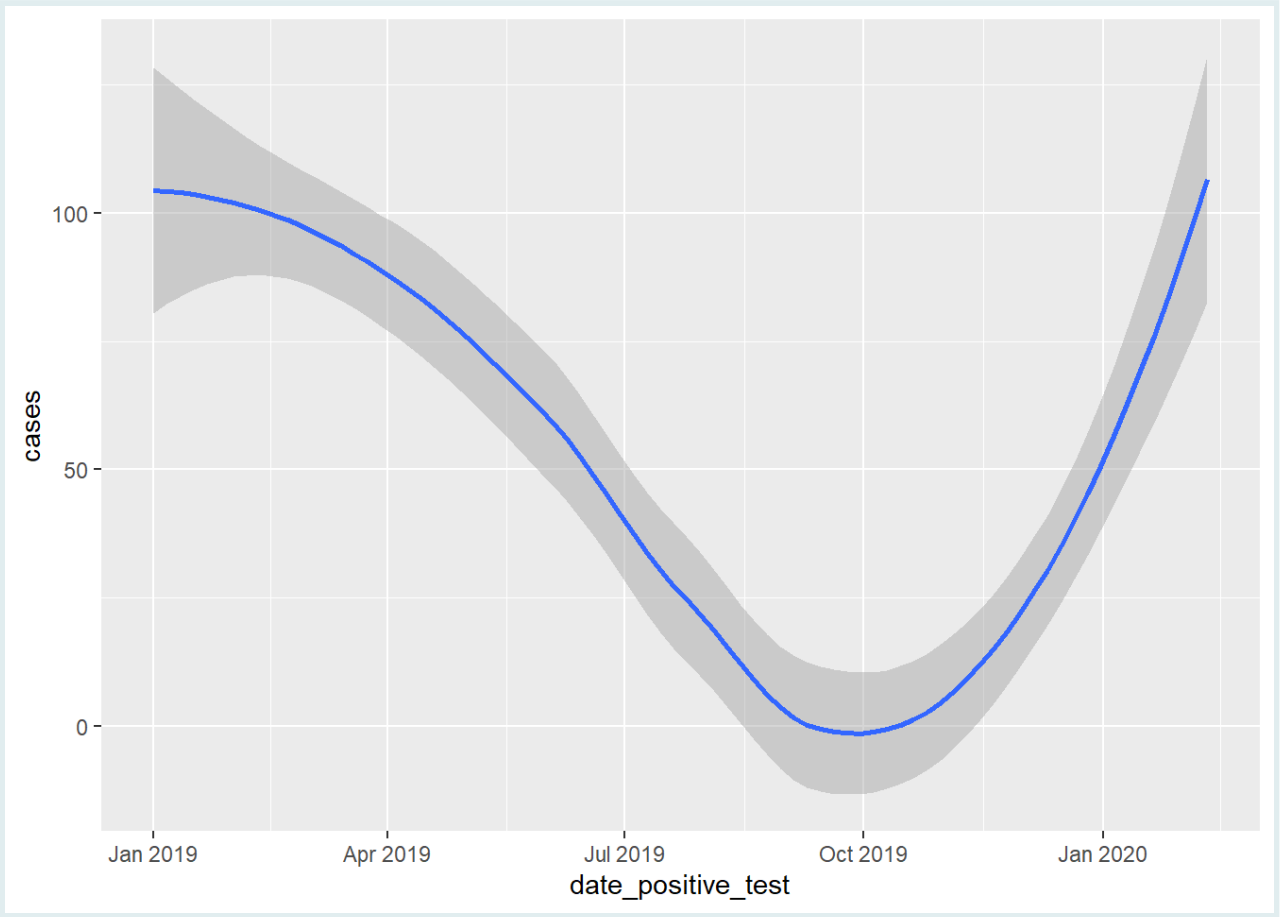


Nous avons une épicourbe valide, mais comme vous pouvez le remarquer, la variabilité quotidienne rend difficile de voir la tendance globale. Lissage à suivre.

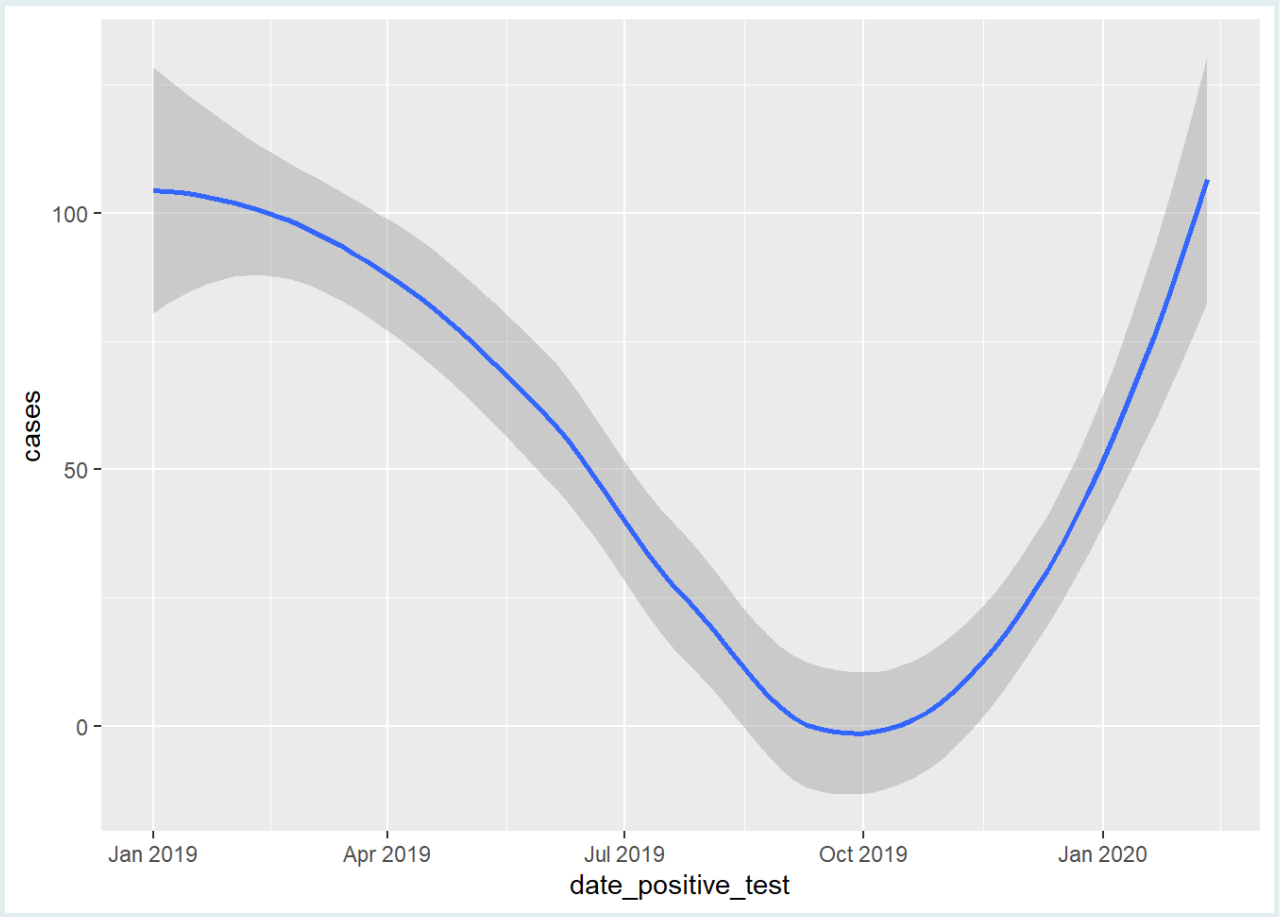
Lissage avec `geom_smooth()`

Une option pour le lissage est la fonction `geom_smooth()`, qui peut effectuer une régression locale avec `loess` pour lisser la série temporelle. Essayons :

```
ggplot(mal_notif_count, aes(x = date_positive_test, y = cases)) +
  geom_smooth()
```



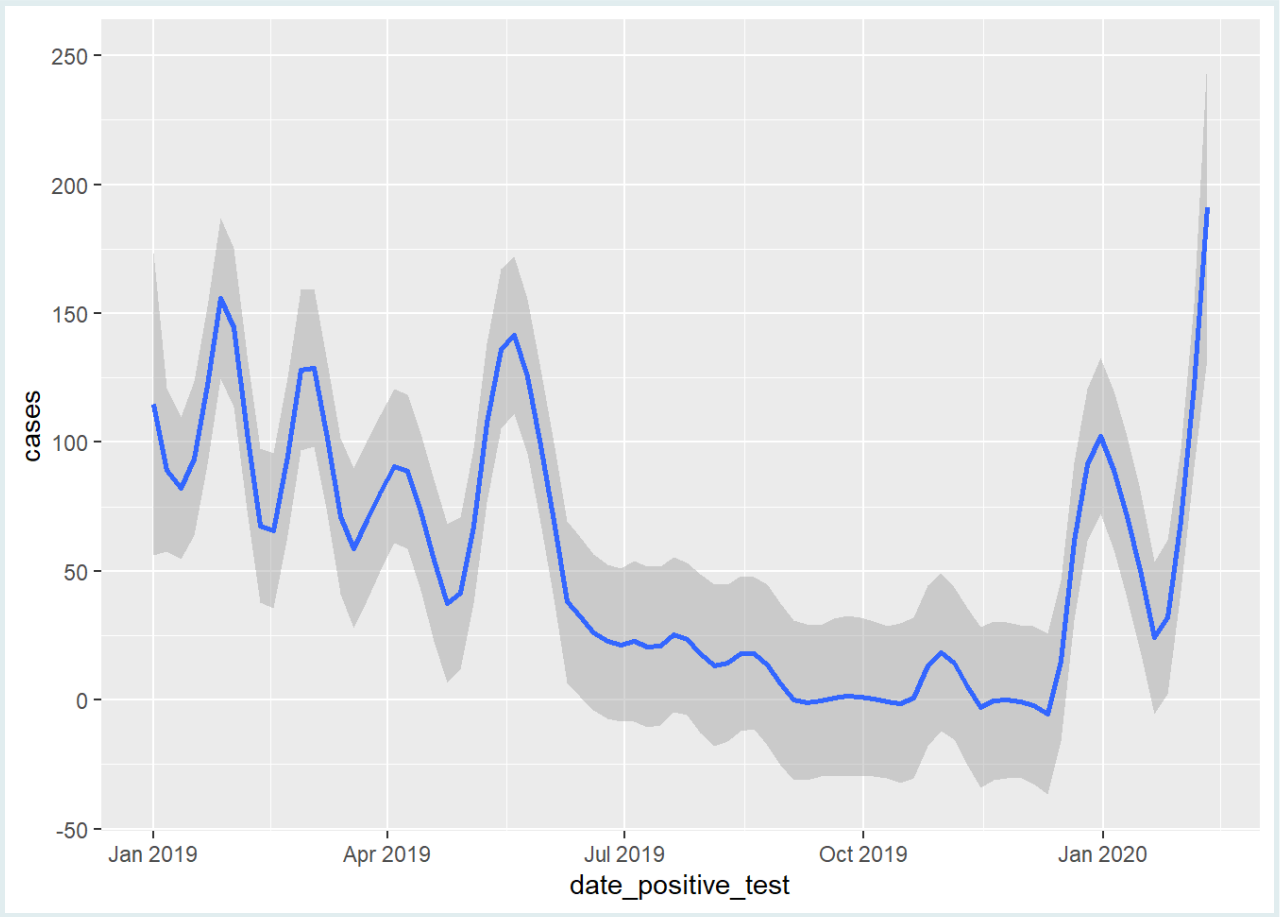

```
# Or we can specify the method explicitly
ggplot(mal_notif_count, aes(x = date_positive_test, y = cases)) +
  geom_smooth(method = "loess")
```



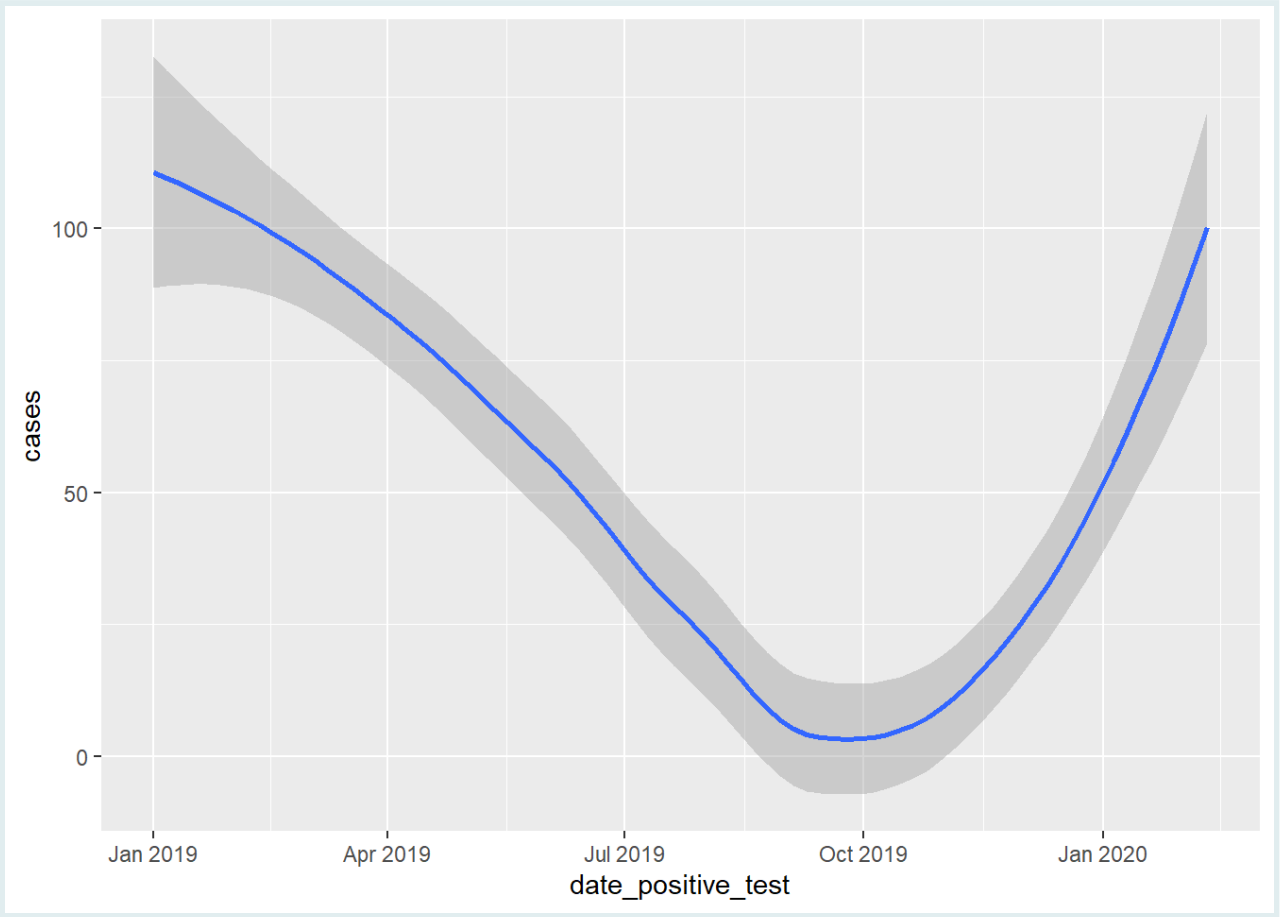
La méthode `loess`, qui signifie “locally weighted scatterplot smoothing”, ajuste une courbe lisse aux données en calculant des moyennes pondérées pour les points proches.

Vous pouvez ajuster la sensibilité du lissage en modifiant l’argument `span`. Un `span` de 0.1 donnera un lissage plus sensible, tandis qu’un `span` de 0.9 donnera un lissage moins sensible.

```
# Ajuster la sensibilité du lissage
ggplot(mal_notif_count, aes(x = date_positive_test, y = cases)) +
  geom_smooth(method = "loess", span = 0.1)
```




```
ggplot(mal_notif_count, aes(x = date_positive_test, y = cases)) +
  geom_smooth(method = "loess", span = 0.9)
```



Lissage par Agrégation

Une autre façon de lisser les données est de les agréger - regrouper les données en intervalles de temps plus grands et calculer des statistiques récapitulatives pour chaque intervalle.

Nous avons déjà vu cela au début de la leçon, lorsque nous avons agrégé les données trimestrielles en données annuelles.

Appliquons-le à nouveau, cette fois en agrégeant l'incidence quotidienne de la malaria en incidence mensuelle. Pour cela, nous utilisons la fonction `floor_date()` du package `lubridate` pour arrondir les dates au mois le plus proche :

```
mal_notif_count %>%
  mutate(month = floor_date(date_positive_test, unit = "month"))
```

```
## # A tibble: 406 × 3
##    date_positive_test cases month
##    <date>             <int> <date>
##  1 2019-01-01            67 2019-01-01
##  2 2019-01-02           120 2019-01-01
##  3 2019-01-03           112 2019-01-01
##  4 2019-01-04           203 2019-01-01
```

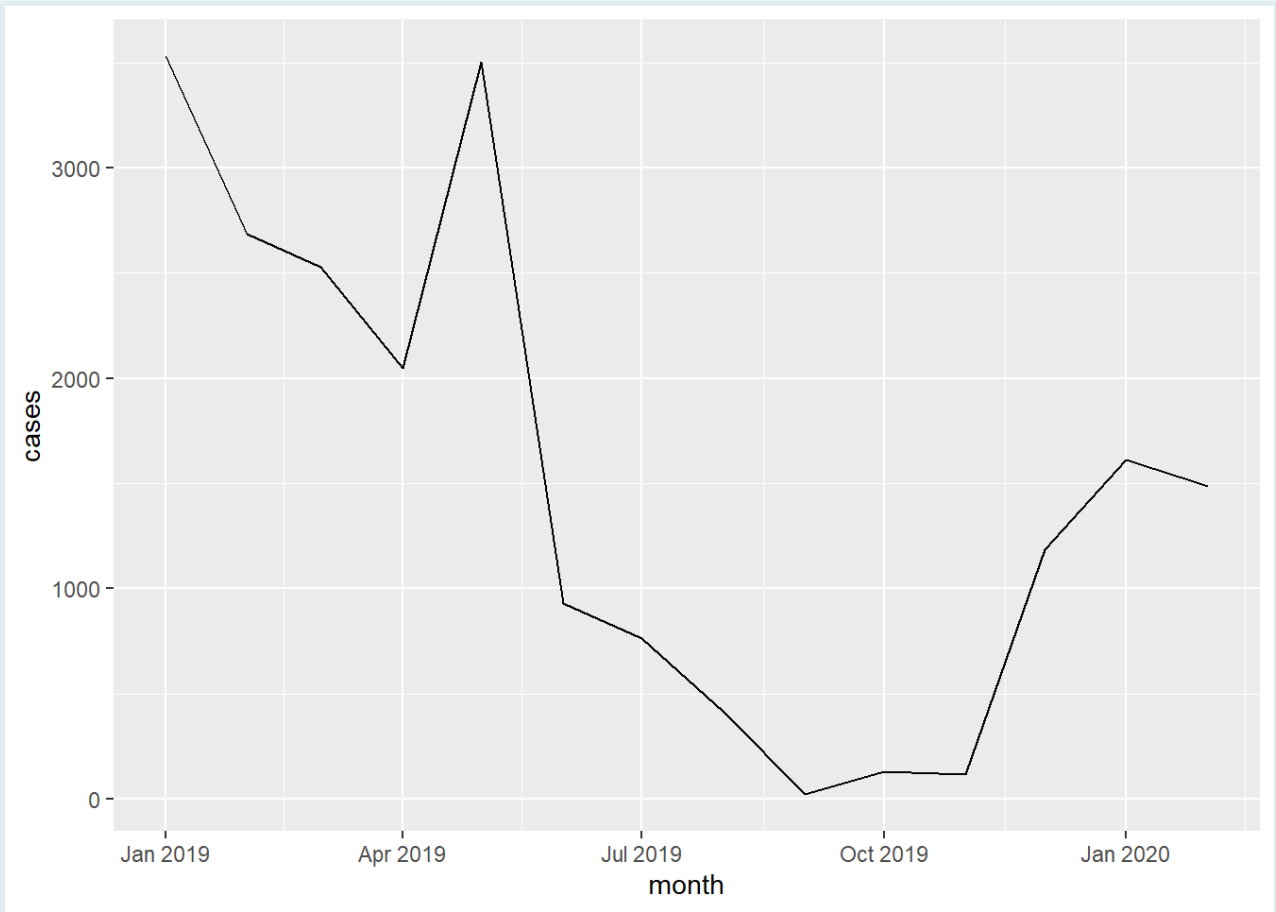
```
##  5 2019-01-05            85 2019-01-01
##  6 2019-01-06             0 2019-01-01
##  7 2019-01-07           115 2019-01-01
##  8 2019-01-08           196 2019-01-01
##  9 2019-01-09             0 2019-01-01
## 10 2019-01-10            89 2019-01-01
## # ℹ 396 more rows
```

Nous pouvons ensuite utiliser `group_by()` et `summarize()` pour calculer le nombre total de cas par mois :

```
mal_monthly <-
  mal_notif_count %>%
  mutate(month = floor_date(date_positive_test, unit = "month")) %>%
  group_by(month) %>%
  summarize(cases = sum(cases))
```

Cela nous donne un tableau d'incidence mensuelle, que nous pouvons tracer pour voir la tendance globale :

```
ggplot(mal_monthly, aes(x = month, y = cases)) +
  geom_line()
```



Voilà ! Une image beaucoup plus claire.

Q : Lissage des Données sur les Décès liés au VIH en Colombie

Considérez ce jeu de données sur les personnes décédées du VIH en Colombie entre 2010 et 2016, provenant de cette URL.

```
colom_hiv_deaths <-
  read_csv(here("data/colombia_hiv_deaths_2010_to_2016.csv")) %>%
  mutate(date_death = ymd(paste(death_year, death_month, death_day, sep = "-
        ")))
colom_hiv_deaths
```

```
## # A tibble: 445 × 26
##    municipality_type death_location  birth_date birth_year
##    <chr>             <chr>           <date>          <dbl>
##  1 Municipal head    Hospital/clinic 1956-05-26       1956
##  2 Municipal head    Hospital/clinic 1983-10-10       1983
##  3 Municipal head    Hospital/clinic 1967-11-22       1967
##  4 Municipal head    Home/address    1964-03-14       1964
##  5 Municipal head    Hospital/clinic 1960-06-27       1960
##  6 Municipal head    Hospital/clinic 1982-03-23       1982
##  7 Municipal head    Hospital/clinic 1964-12-09       1964
##  8 Municipal head    Hospital/clinic 1975-01-15       1975
##  9 Municipal head    Hospital/clinic 1988-02-15       1988
## 10 Municipal head    Hospital/clinic NA                 NA
##    birth_month birth_day death_year death_month death_day
##    <chr>           <dbl>      <dbl> <chr>           <dbl>
##  1 May                26       2012 Sep                14
##  2 Oct                10       2012 Mar                17
##  3 Nov                22       2011 Oct                19
##  4 Mar                14       2012 Nov                19
##  5 Jun                27       2012 Jan                13
##  6 Mar                23       2013 Dec                25
##  7 Dec                 9       2013 Feb                21
##  8 Jan                15       2013 Dec                 6
##  9 Feb                15       2013 Nov                 2
## 10 <NA>               NA       2013 Oct                 6
## # ℹ 435 more rows
## # ℹ 17 more variables: age_at_death <dbl>, gender <chr>, …
```

En utilisant les étapes enseignées ci-dessus :

1. Créez un tableau qui compte les décès liés au VIH par mois.
2. Tracez une épicourbe des décès par mois
3. Appliquez `geom_smooth` à l'épicourbe pour une visualisation plus lisse. Assurez-vous de choisir une portée appropriée pour le lissage.

Lissage avec des Moyennes Mobiles

Une autre technique pour lisser les données bruyantes de séries temporelles est de calculer des **moyennes mobiles**. Cela prend la moyenne d'un nombre fixe de

points, centrée autour de chaque point de données.

La fonction `rollmean()` du package {zoo} sera votre principal outil pour calculer les moyennes mobiles.

Appliquons une moyenne mobile sur 14 jours pour lisser nos données quotidiennes de cas de malaria :

```
mal_notif_count <- mal_notif_count %>%
  mutate(roll_cases = rollmean(cases, k = 14, fill = NA))
mal_notif_count
```

```
## # A tibble: 406 × 3
##    date_positive_test cases roll_cases
##    <date>             <int>      <dbl>
##  1 2019-01-01            67       NA
##  2 2019-01-02           120       NA
##  3 2019-01-03           112       NA
##  4 2019-01-04           203       NA
##  5 2019-01-05            85       NA
##  6 2019-01-06             0       NA
##  7 2019-01-07           115       83.4
##  8 2019-01-08           196       82.9
##  9 2019-01-09             0       79.3
## 10 2019-01-10            89       82.1
## # ℹ 396 more rows
```

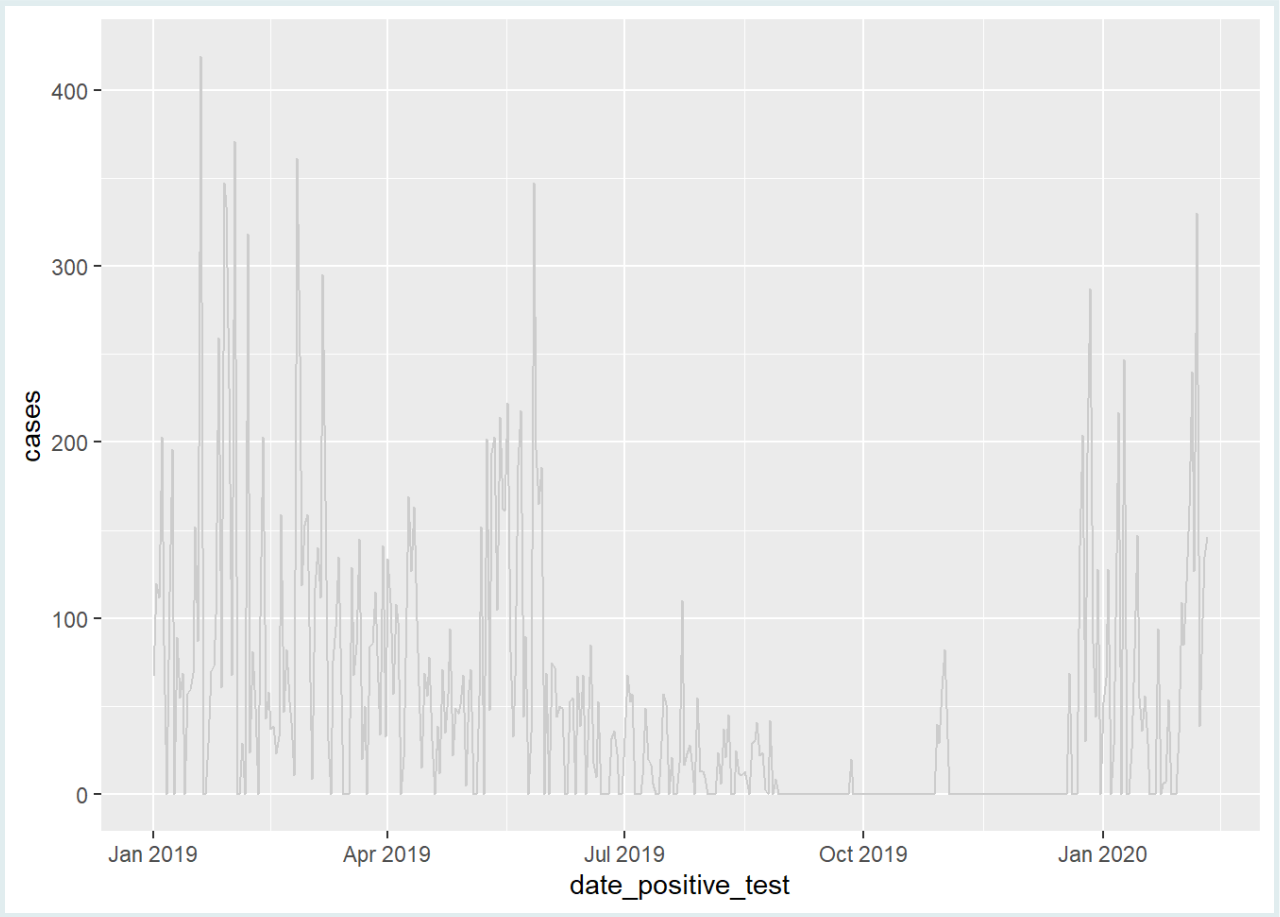
Les arguments clés sont :

- `x` : La série temporelle à lisser
- `k` : Le nombre de points avant et après pour faire la moyenne
- `fill` : Comment gérer les données manquantes dans chaque fenêtre

Cela calcule la moyenne mobile sur 14 jours, laissant les données manquantes comme NA. Notez que les 6 premiers jours sont NA, car il n'y a pas assez de points pour faire une moyenne (avec un `k` de 14, nous avons besoin de 7 jours avant et 7 jours après chaque point pour calculer la moyenne mobile.

Tracer le graphique :

```
mal_notif_count %>%
  ggplot(aes(x = date_positive_test, y = cases)) +
  geom_line(color = "gray80")
```



Souvent, on vous demandera de tracer une moyenne mobile des *dernières* 1 ou 2 semaines. Pour cela, vous devez régler l'argument `align` sur `"right"` :

```
mal_notif_count_right <-
  mal_notif_count %>%
  mutate(roll_cases = rollmean(cases, k = 14, fill = NA, align = "right"))

head(mal_notif_count_right, 15)
```

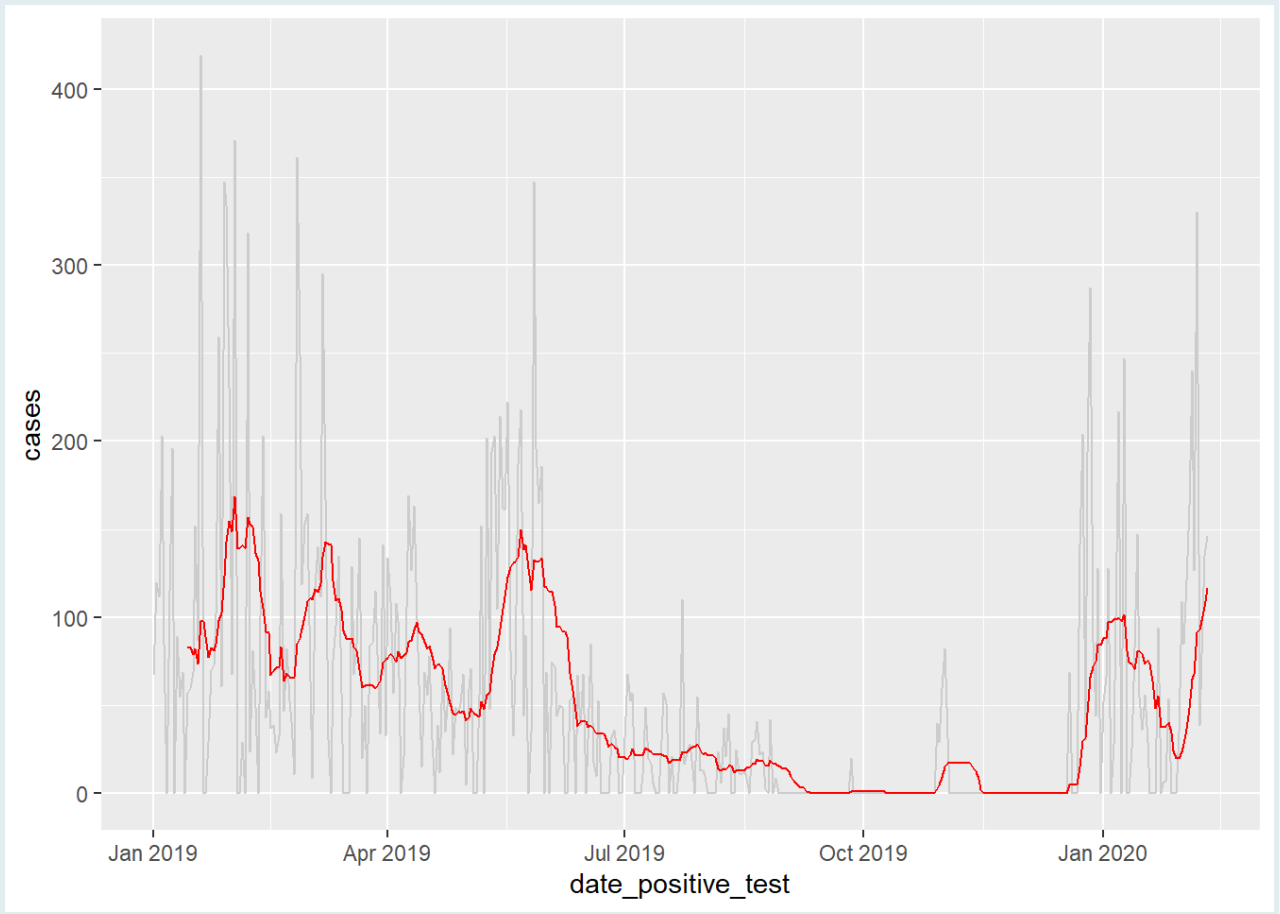
```
## # A tibble: 15 × 3
##    date_positive_test cases roll_cases
##    <date>             <int>      <dbl>
##  1 2019-01-01            67         NA
##  2 2019-01-02           120         NA
##  3 2019-01-03           112         NA
##  4 2019-01-04           203         NA
##  5 2019-01-05            85         NA
##  6 2019-01-06             0         NA
##  7 2019-01-07           115         NA
##  8 2019-01-08           196         NA
##  9 2019-01-09             0         NA
## 10 2019-01-10            89         NA
## 11 2019-01-11            55         NA
## 12 2019-01-12            69         NA
## 13 2019-01-13             0         NA
```

```
## 14 2019-01-14            57       83.4
## 15 2019-01-15            60       82.9
```

Notez que maintenant les 13 premiers jours sont NA, car il n'y a pas assez de points pour faire une moyenne. C'est parce que nous calculons la moyenne des *14 derniers jours*, et les 13 premiers jours n'ont pas 14 jours avant eux.

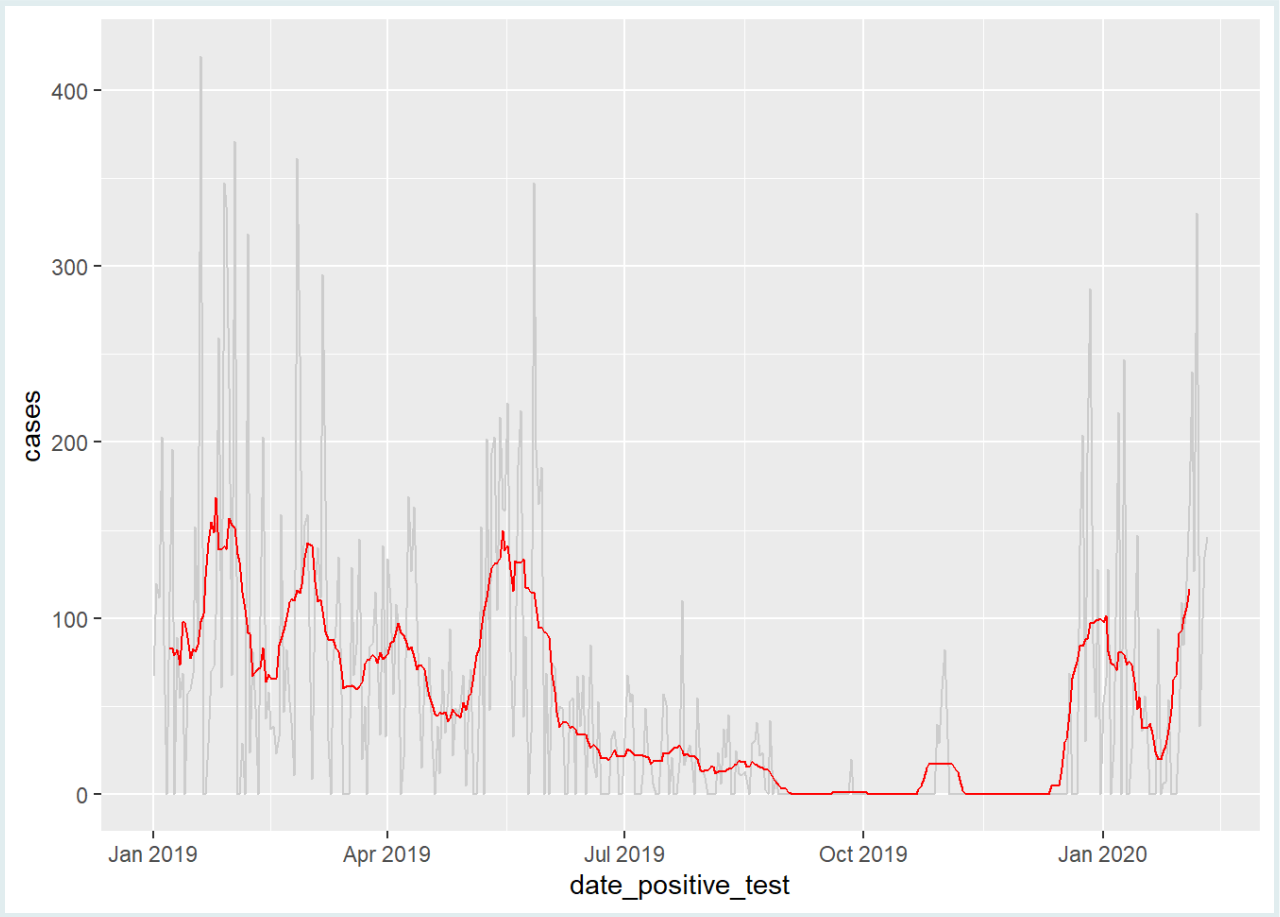
La sortie ne change pas beaucoup dans ce cas :

```
ggplot(mal_notif_count_right, aes(x = date_positive_test, y = cases)) +
  geom_line(color = "gray80") +
  geom_line(aes(y = roll_cases), color = "red")
```



Finalement, traçons à la fois les données originales et lissées :

```
mal_notif_count %>%
  ggplot(aes(x = date_positive_test, y = cases)) +
  geom_line(color = "gray80") +
  geom_line(aes(y = roll_cases), color = "red")
```



En résumé, la fonction `rollmean()` nous permet de calculer facilement une moyenne mobile sur une fenêtre fixe pour lisser et mettre en évidence les motifs dans les données bruyantes de séries temporelles.

Q : Lissage avec des moyennes mobiles

Considérez à nouveau le jeu de données sur les décès de patients VIH en Colombie :


```
colom_hiv_deaths
```

```
## # A tibble: 445 × 26
##    municipality_type death_location  birth_date birth_year
##    <chr>             <chr>           <date>          <dbl>
##  1 Municipal head    Hospital/clinic 1956-05-26       1956
##  2 Municipal head    Hospital/clinic 1983-10-10       1983
##  3 Municipal head    Hospital/clinic 1967-11-22       1967
##  4 Municipal head    Home/address    1964-03-14       1964
```

```
##  7 Municipal head    Hospital/clinic 1964-12-09       1964
##  8 Municipal head    Hospital/clinic 1975-01-15       1975
##  9 Municipal head    Hospital/clinic 1988-02-15       1988
## 10 Municipal head    Hospital/clinic NA                 NA
##    birth_month birth_day death_year death_month death_day
##    <chr>           <dbl>      <dbl> <chr>           <dbl>
##  1 May                26       2012 Sep                14
##  2 Oct                10       2012 Mar                17
##  3 Nov                22       2011 Oct                19
##  4 Mar                14       2012 Nov                19
##  5 Jun                27       2012 Jan                13
##  6 Mar                23       2013 Dec                25
##  7 Dec                 9       2013 Feb                21
##  8 Jan                15       2013 Dec                 6
##  9 Feb                15       2013 Nov                 2
## 10 <NA>               NA       2013 Oct                 6
## # ℹ 435 more rows
## # ℹ 17 more variables: age_at_death <dbl>, gender <chr>, …
```

Le code suivant calcule le nombre de décès par jour :


```
colom_hiv_deaths_per_day <-
  colom_hiv_deaths %>%
  group_by(date_death) %>%
  summarize(deaths = n()) %>%
  complete(date_death = seq.Date(min(date_death),
                                 max(date_death),
                                 by = "day"),
           fill = list(deaths = 0))

colom_hiv_deaths_per_day
```

```
## # A tibble: 2,543 × 2
##    date_death deaths
##    <date>      <int>
##  1 2010-01-05      1
##  2 2010-01-06      0
##  3 2010-01-07      0
##  4 2010-01-08      0
##  5 2010-01-09      1
##  6 2010-01-10      0
##  7 2010-01-11      1
##  8 2010-01-12      0
##  9 2010-01-13      0
## 10 2010-01-14      0
## # ℹ 2,533 more rows
```

Votre tâche est de créer une nouvelle colonne qui calcule la moyenne mobile des décès par jour sur une période de 14 jours. Ensuite, tracez cette moyenne mobile aux côtés des données brutes.

Axes Secondaires

Comprendre le Concept d'un Axe Y Secondaire

Un axe y secondaire est utile lorsque l'on souhaite visualiser deux mesures différentes avec des échelles distinctes sur le même graphique. Cette approche est utile lorsque les variables ont des unités ou des magnitudes différentes, rendant la comparaison directe sur une seule échelle difficile.

Bien que certains experts en visualisation de données déconseillent l'utilisation d'axes secondaires, les décideurs en santé publique apprécient souvent ces graphiques.

Création d'un Graphique avec un Axe Y Secondaire

Démontrons comment créer un graphique avec un axe y secondaire en utilisant notre jeu de données sur les notifications de malaria :

Étape 1 : Créer des Comptes Cumulatifs de Cas

D'abord, nous allons agréger nos données sur la malaria pour calculer les comptes cumulatifs de cas.

```
mal_notif_count_cumul <-
  mal_notif_count %>%
  mutate(cumul_cases = cumsum(cases))

mal_notif_count_cumul
```

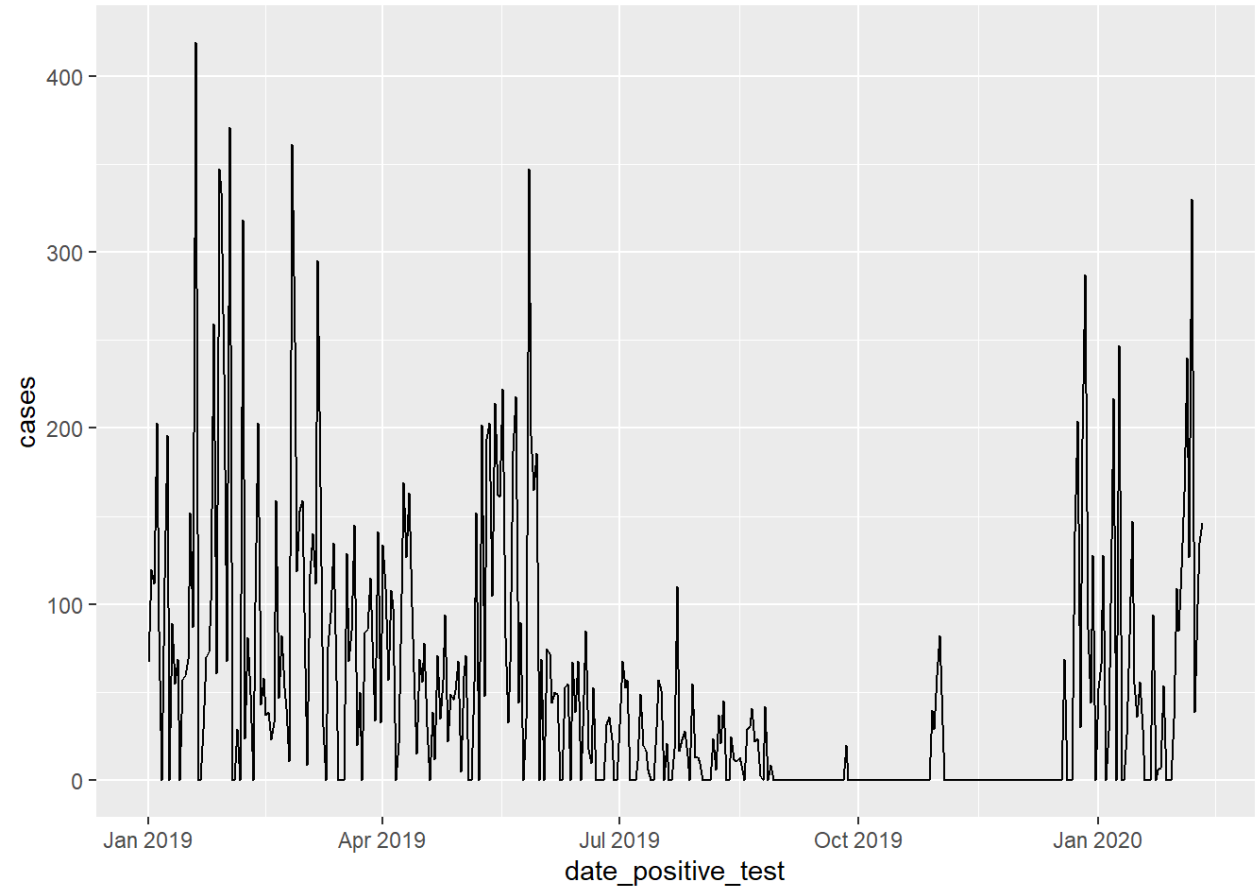
```
## # A tibble: 406 × 4
##    date_positive_test cases roll_cases cumul_cases
##    <date>             <int>      <dbl>       <int>
##  1 2019-01-01            67       NA            67
##  2 2019-01-02           120       NA           187
##  3 2019-01-03           112       NA           299
##  4 2019-01-04           203       NA           502
##  5 2019-01-05            85       NA           587
##  6 2019-01-06             0       NA           587
##  7 2019-01-07           115       83.4         702
##  8 2019-01-08           196       82.9         898
##  9 2019-01-09             0       79.3         898
```

```
## 10 2019-01-10            89       82.1         987
## # ℹ 396 more rows
```

Étape 2 : Identifier le Besoin d'un Axe Y Secondaire

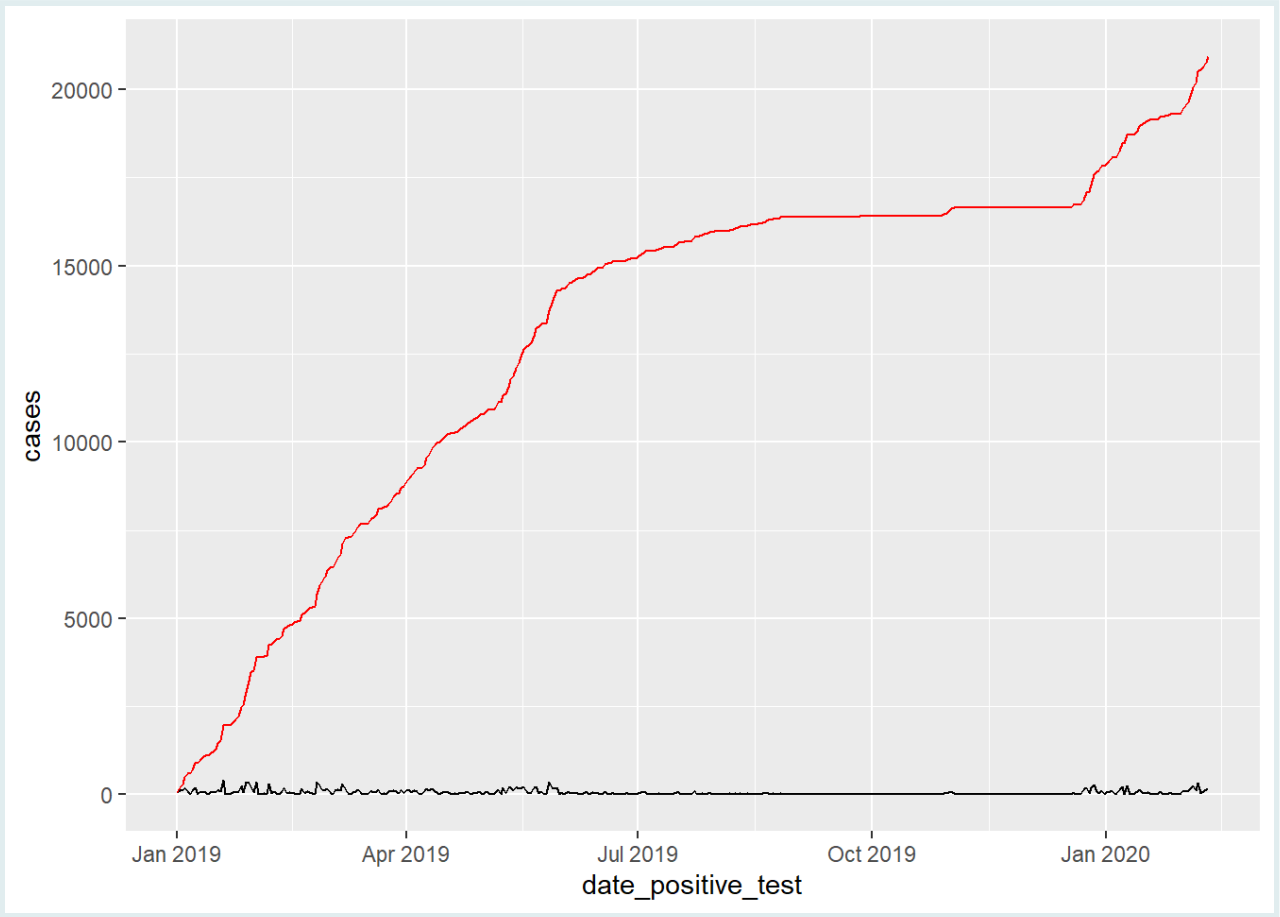
Maintenant, nous pouvons commencer à tracer. D'abord, nous traçons seulement les cas quotidiens :

```
# Tracer les cas totaux de malaria
ggplot(mal_notif_count_cumul, aes(x = date_positive_test)) +
  geom_line(aes(y = cases))
```



Si nous essayons d'ajouter des cas cumulatifs sur le même axe y, les cas quotidiens seront éclipsés et leur magnitude sera difficile à lire en raison de l'échelle beaucoup plus grande des données cumulatives :

```
# Ajouter des cas cumulatifs de malaria au graphique
ggplot(mal_notif_count_cumul, aes(x = date_positive_test)) +
  geom_line(aes(y = cases)) +
  geom_line(aes(y = cumul_cases), color = "red")
```



Pour afficher efficacement les deux ensembles de données, nous devons introduire un axe y secondaire.

Étape 3 : Calcul et Application du Facteur d'Échelle

Avant d'ajouter un axe y secondaire, nous devons déterminer un *facteur d'échelle* en comparant les gammes de cas et de cas cumulatifs.

Le facteur d'échelle est généralement le rapport des valeurs maximales des deux ensembles de données. Voyons quelles sont les valeurs maximales pour chaque variable :

```
max(mal_notif_count_cumul$cases)
```

```
## [1] 419
```

```
max(mal_notif_count_cumul$cumul_cases)
```

```
## [1] 20939
```

Avec un maximum d'environ 20000 pour les cas cumulatifs, et environ 400 pour les cas quotidiens, nous pouvons voir que les cas cumulatifs sont environ 50 fois plus importants que les cas quotidiens, donc notre facteur d'échelle sera d'environ 50.

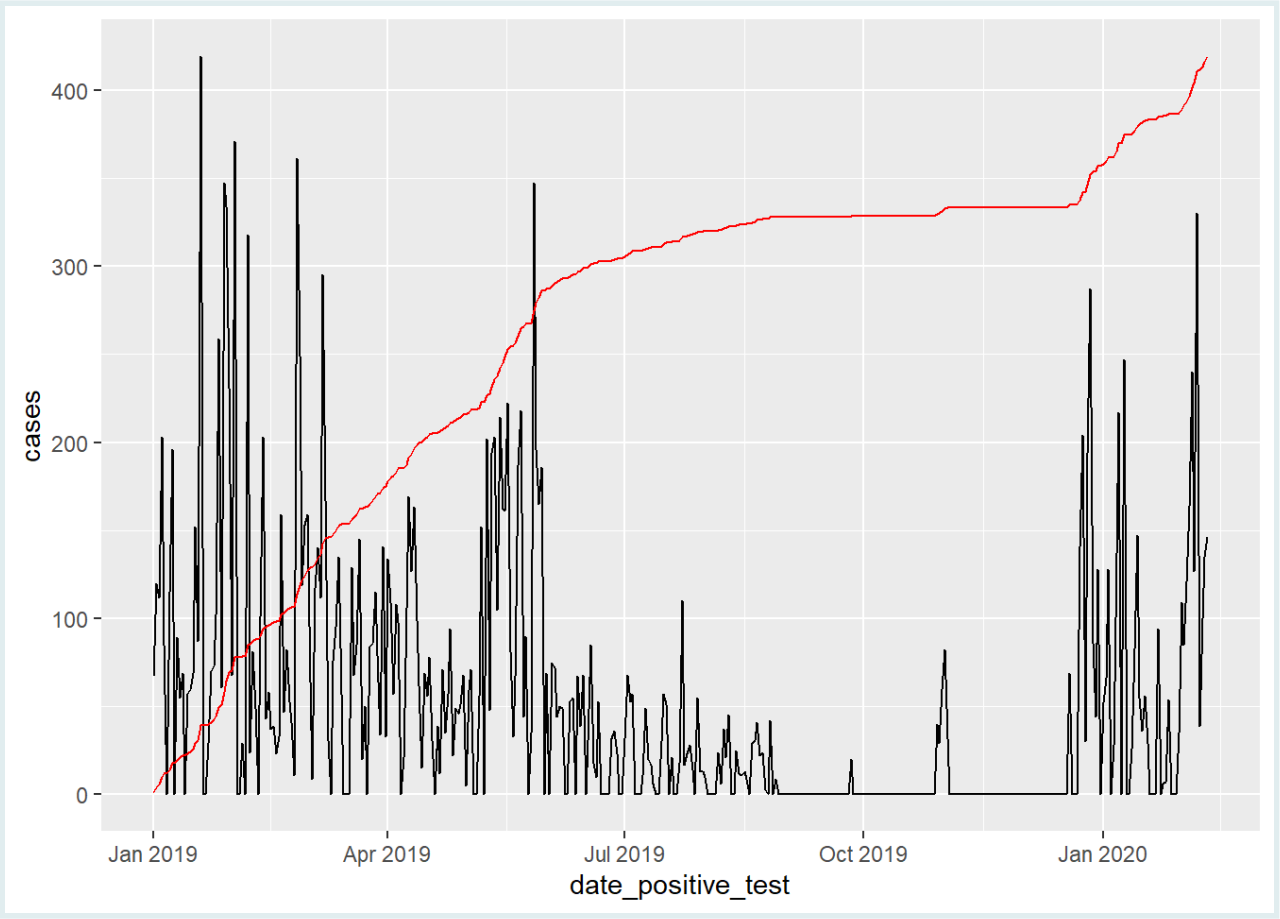
Plus précisément, le facteur d'échelle sera :

```
max(mal_notif_count_cumul$cumul_cases) / max(mal_notif_count_cumul$cases)
```

```
## [1] 49.97375
```

Nous devrons diviser les cas cumulatifs par ce ratio pour forcer les deux variables à être sur une échelle similaire :

```
ggplot(mal_notif_count_cumul, aes(x = date_positive_test)) +
  geom_line(aes(y = cases)) +
  geom_line(aes(y = cumul_cases / 49.97), color = "red") # diviser par le
        facteur d'échelle
```



Super ! Maintenant, nous pouvons clairement voir les deux ensembles de données sur le même graphique, et leurs points maximaux sont alignés. Cependant, l'axe y n'est plus pertinent pour la ligne des cas cumulatifs en rouge. Nous devons ajouter un axe y secondaire pour cela.

Normalement, vous assigneriez le facteur d'échelle à une variable, puis utiliseriez cette variable dans la fonction `geom_line()`. Dans ce cas, nous l'écrivons directement dans la fonction pour une meilleure compréhension.

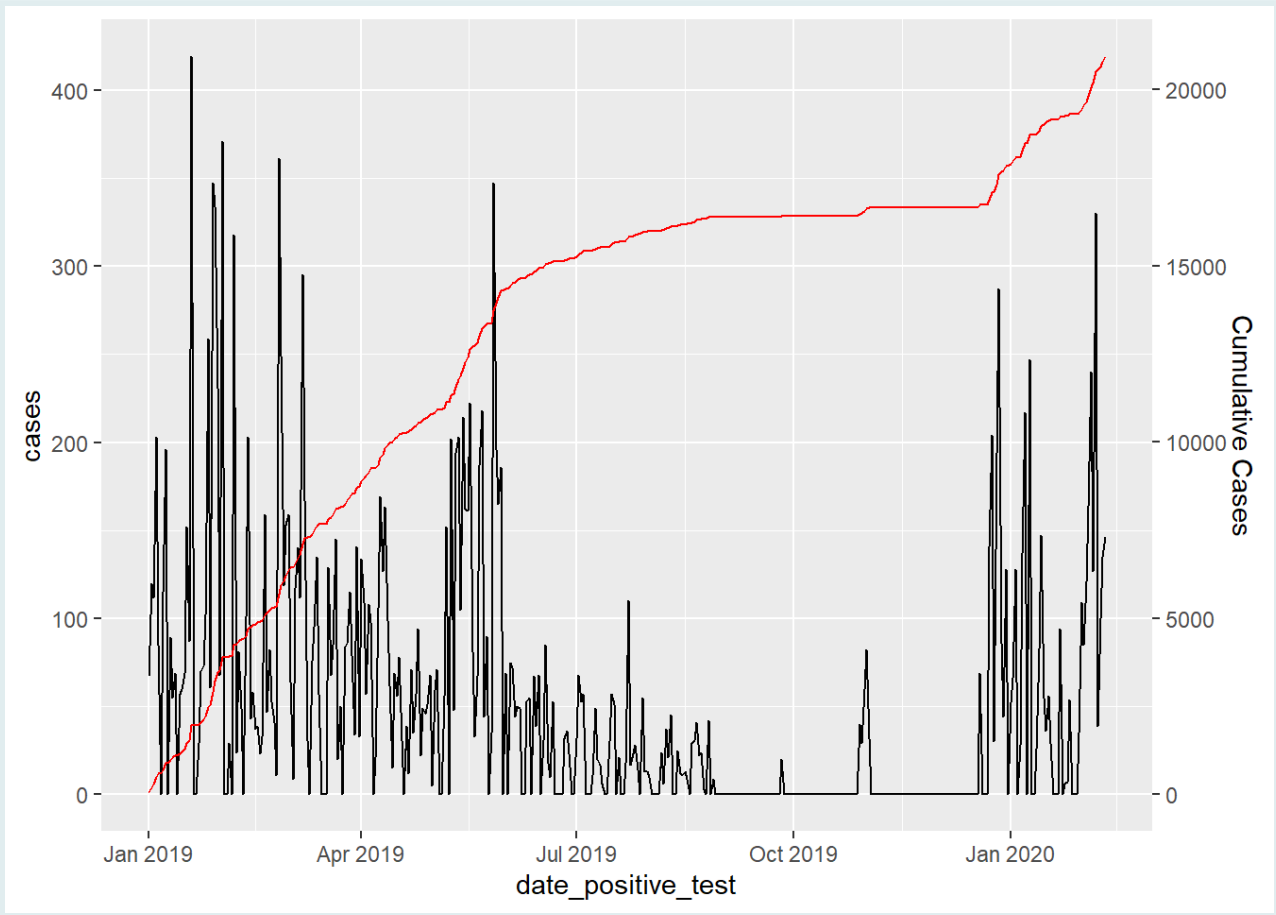
Étape 4 : Ajout de l'Axe Y Secondaire

Nous utiliserons la fonction `sec_axis()` de {ggplot2}. Les arguments clés sont `trans`, qui indique combien multiplier ou diviser l'axe y original, et `name`, qui spécifie le nom de l'axe secondaire.

Dans notre cas, nous voulons que l'axe secondaire soit environ 49.97 fois plus grand que l'axe original, donc nous utiliserons `trans = ~ .x * 49.97`. (Le symbole `~` est un opérateur spécial qui indique à R de traiter l'expression qui suit comme une fonction, dont l'entrée est indiquée par le symbole `.x`.)

Mettre cela en œuvre :

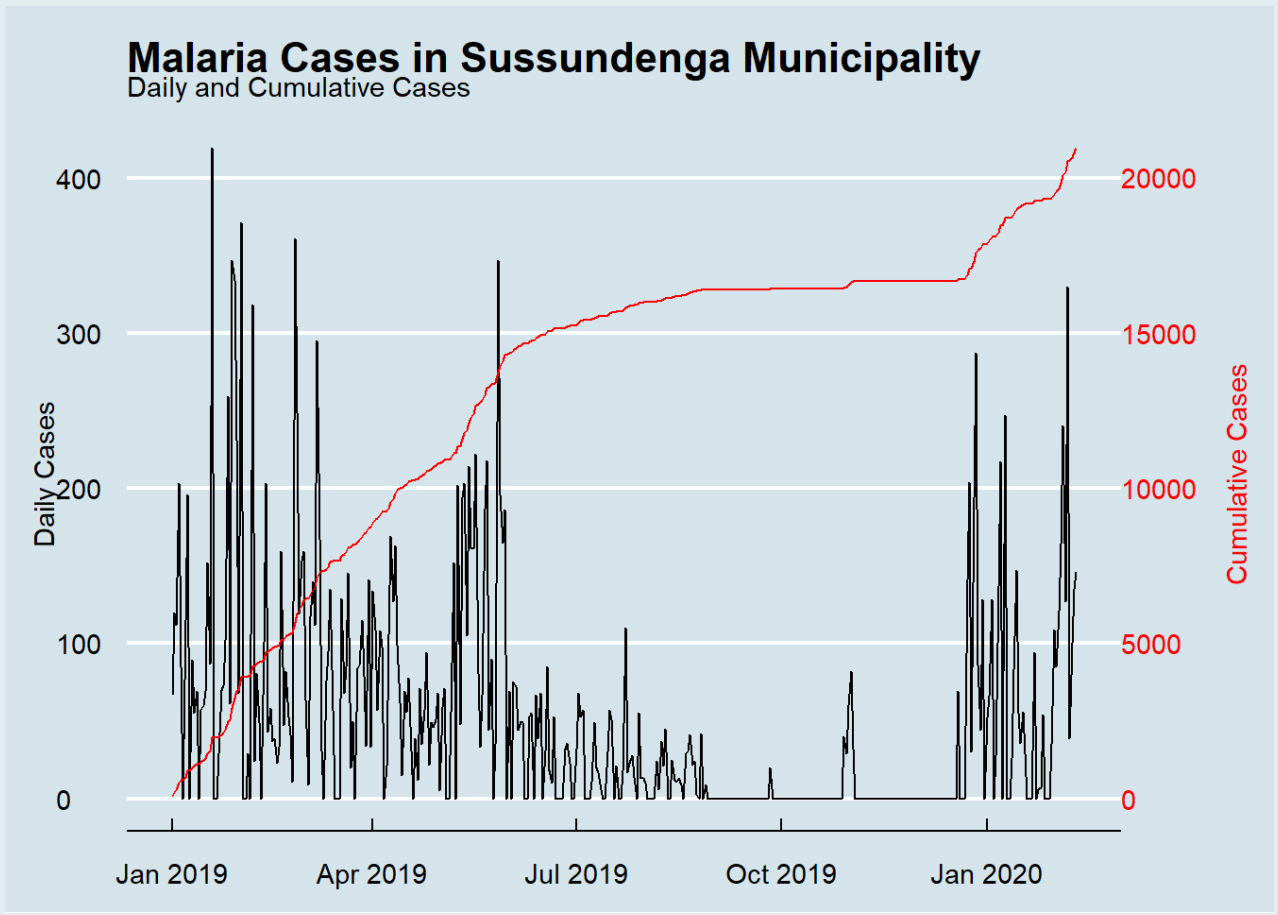
```
# Ajouter un axe y secondaire
ggplot(mal_notif_count_cumul, aes(x = date_positive_test)) +
  geom_line(aes(y = cases)) +
  geom_line(aes(y = cumul_cases / 49.97), color = "red") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~ .x * 49.97,
                                         name = "Cumulative Cases"))
```



Étape 5 : Améliorer la Lisibilité du Graphique

Pour améliorer la lisibilité, nous allons faire correspondre les étiquettes de l'axe secondaire en rouge, assorties à la couleur de la ligne des cas cumulatifs, et nous ajouterons un peu de formatage supplémentaire au graphique :

```
# Finaliser le graphique avec des axes coordonnés en couleur
ggplot(mal_notif_count_cumul, aes(x = date_positive_test)) +
  geom_line(aes(y = cases)) +
  geom_line(aes(y = cumul_cases / 49.97), color = "red") +
  scale_y_continuous(
    name = "Daily Cases",
    sec.axis = sec_axis(~ . * 49.97, name = "Cumulative Cases")
  ) +
  labs(title = "Malaria Cases in Sussundenga Municipality",
       subtitle = "Daily and Cumulative Cases",
       x = NULL) +
  theme_economist() +
  theme(axis.text.y.right = element_text(color = "red"),
        axis.title.y.right = element_text(color = "red"))
```



Voilà ! Nous avons avec succès ajouté un axe y secondaire à un graphique, permettant la comparaison de deux ensembles de données avec des échelles différentes dans une seule visualisation.

Q : Axes secondaires

Revenez sur le jeu de données `colom_hiv_deaths_per_day`.

```
colom_hiv_deaths_per_day
```

```
## # A tibble: 2,543 × 2
##    date_death deaths
##    <date>      <int>
##  1 2010-01-05      1
##  2 2010-01-06      0
##  3 2010-01-07      0
##  4 2010-01-08      0
##  5 2010-01-09      1
##  6 2010-01-10      0
##  7 2010-01-11      1
##  8 2010-01-12      0
##  9 2010-01-13      0
```

```
## 10 2010-01-14      0
## # ℹ 2,533 more rows
```

Votre tâche est de créer un graphique avec deux axes y : un pour les décès quotidiens et un autre pour les décès cumulatifs en Colombie.

Conclusion !

- Ce cours a fourni une introduction complète à la visualisation des données de séries temporelles à l'aide de graphiques en ligne, en mettant l'accent sur les applications pratiques dans les contextes de santé publique. En maîtrisant ces techniques, vous êtes maintenant équipé pour communiquer efficacement les tendances des données complexes et informer la prise de décision basée sur les données dans les domaines de la santé et connexes.*

‣ Visualiser les notifications de TB au fil du temps.

‣ Surmonter les défis de la conversion des données trimestrielles en format annuel.

‣ Créer des Graphiques en Ligne Basiques et Groupés en employant des méthodes de remodelage des données comme `pivot_longer()`.

‣ Améliorer les tracés avec des techniques de gestion des étiquettes et de personnalisation des couleurs comme `ggrepel::geom_text_repel()` pour une clarté dans le placement des étiquettes.

‣ Apprendre l'importance de représenter la variabilité à travers des intervalles de confiance dans les données avec `geom_ribbon()`.

‣ Lisser les données bruyantes avec `geom_smooth()` et des moyennes mobiles pour une visualisation des tendances plus claire.

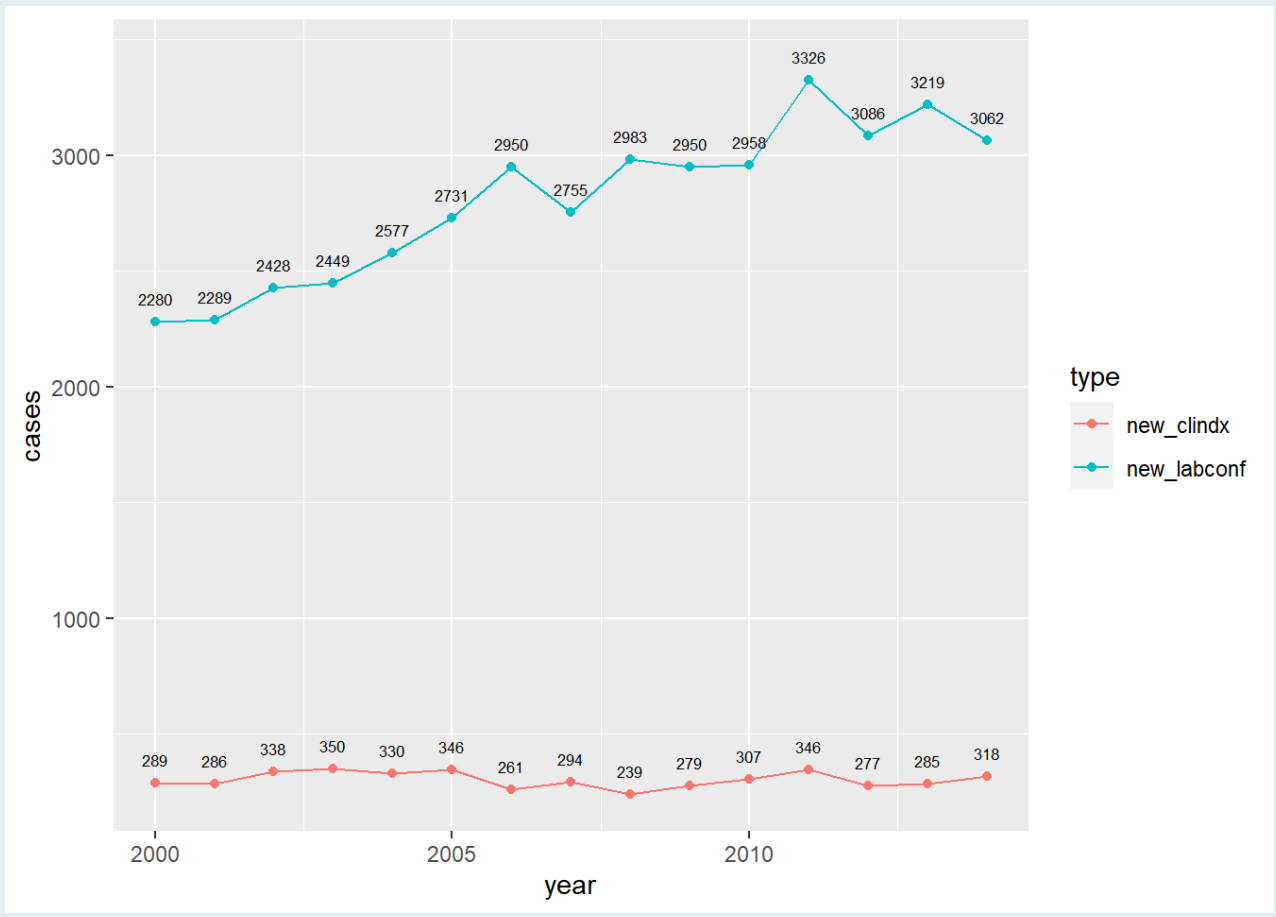
‣ Maîtriser l'utilisation d'un axe y secondaire pour comparer des mesures distinctes.

Prochaines Étapes : Continuez à explorer des techniques de visualisation avancées et appliquez ces compétences à divers ensembles de données pour approfondir votre compréhension et votre maîtrise de l'analyse et de la visualisation des données.

Solutions

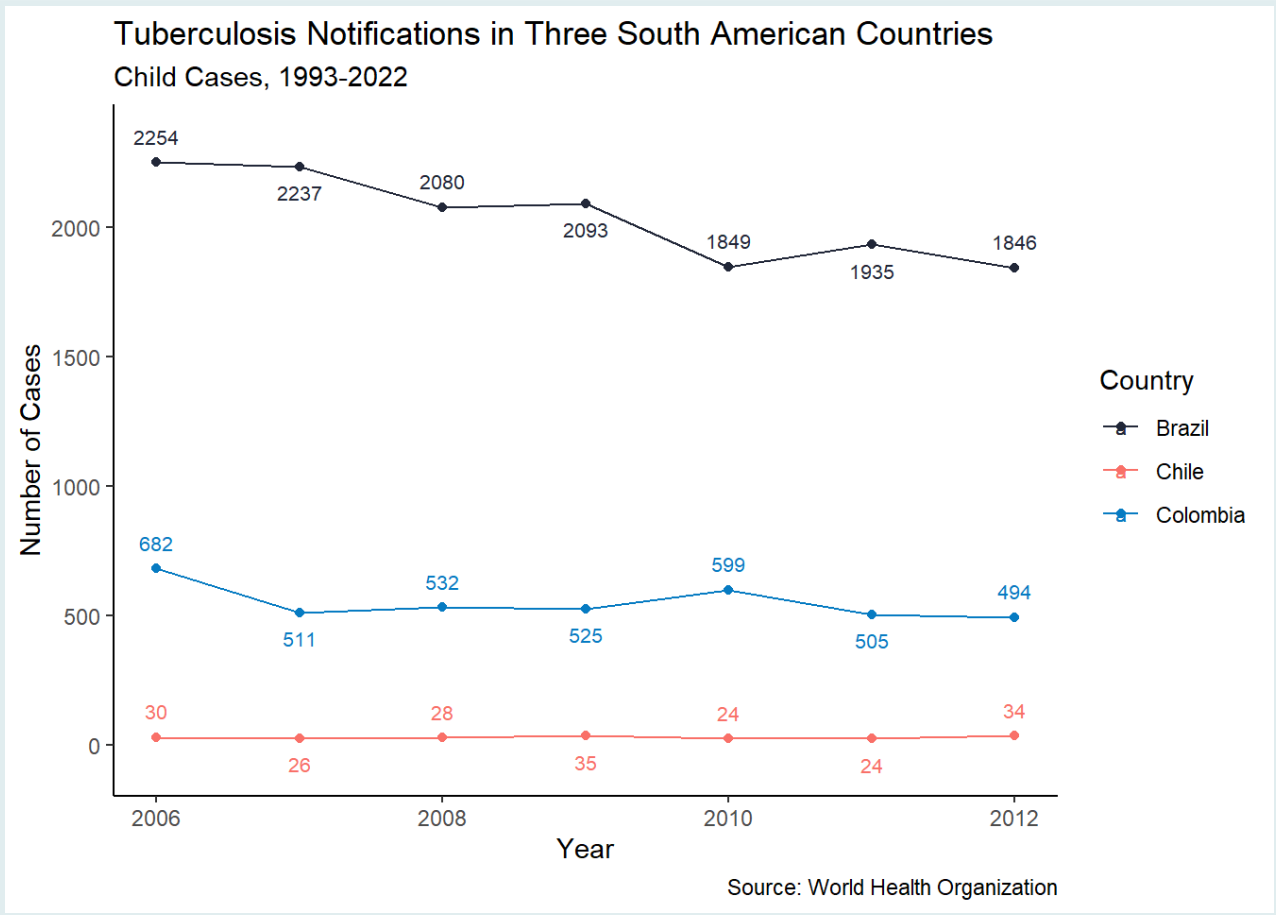
Q: Mise en forme et représentation des données sur la tuberculose

```
tb_benin_long <- tb_data_benin %>%
  pivot_longer(cols = c("new_clindx", "new_labconf")) %>%
  rename(type = name, cases = value)

ggplot(tb_benin_long, aes(x = year, y = cases, colour = type, group = type)) +
  geom_line() +
  geom_point() +
  geom_text(aes(label = cases), size = 2.2, nudge_y = 100, color = "black")
```



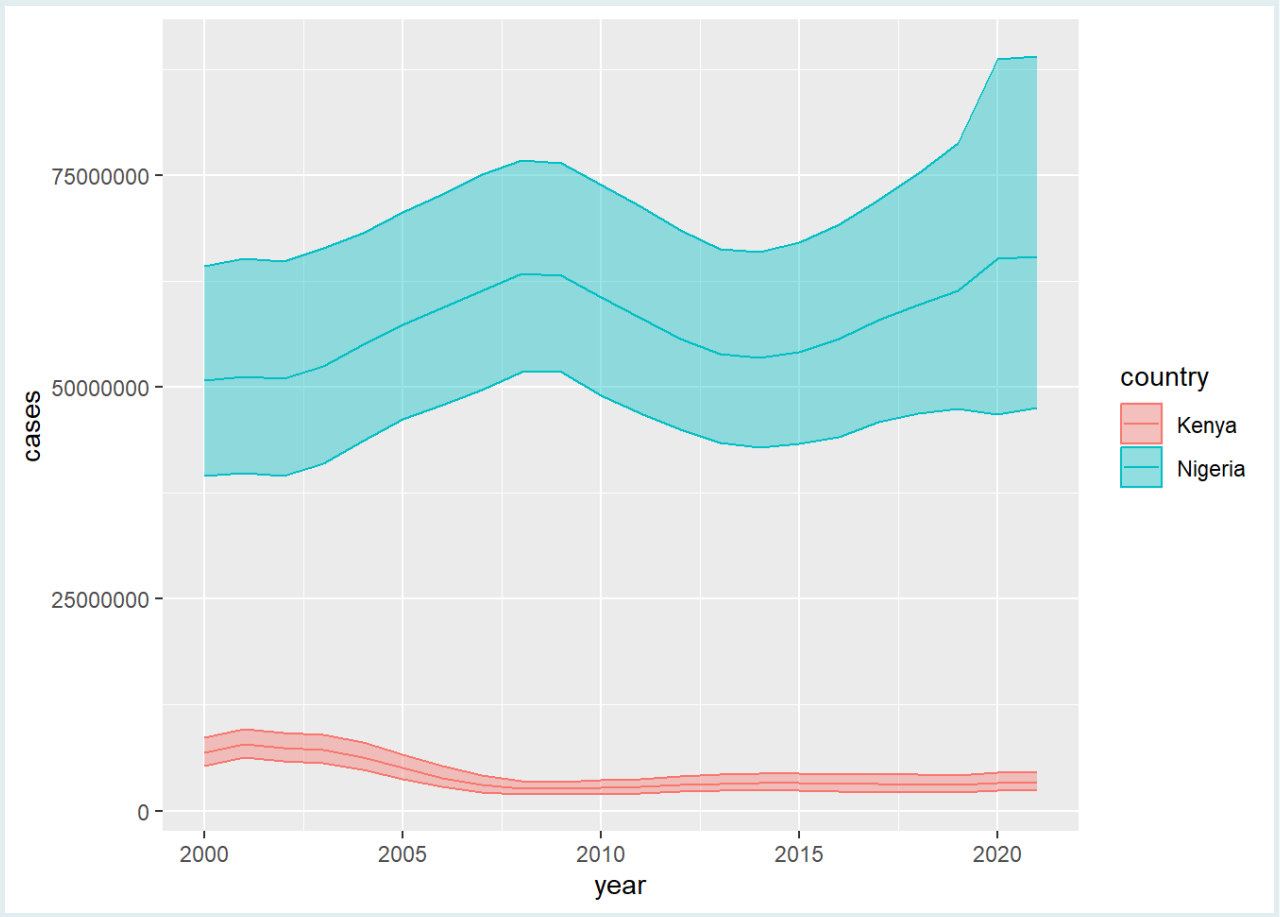
Q: Améliorations esthétiques

```
even_years_southam <- tb_child_cases_southam %>%
  filter(year %% 2 == 0) # Keep only years that are multiples of 2

odd_years_southam <- tb_child_cases_southam %>%
  filter(year %% 2 == 1) # Keep only years that are not multiples of 2


tb_child_cases_southam %>%
  ggplot(aes(x = year, y = tb_cases_children, color = country)) +
  geom_line() +
  geom_point() +
  geom_text(data = even_years_southam, aes(label = tb_cases_children),
            nudge_y = 100, size = 2.8) +
  geom_text(data = odd_years_southam, aes(label = tb_cases_children),
            nudge_y = -100, size = 2.8) +
  scale_color_manual(values = c("#212738", "#F97068", "#067BC2")) +
  labs(title = "Tuberculosis Notifications in Three South American Countries",
       subtitle = "Child Cases, 1993-2022",
       caption = "Source: World Health Organization",
       x = "Year",
       y = "Number of Cases",
       color = "Country") +
  theme_classic()
```



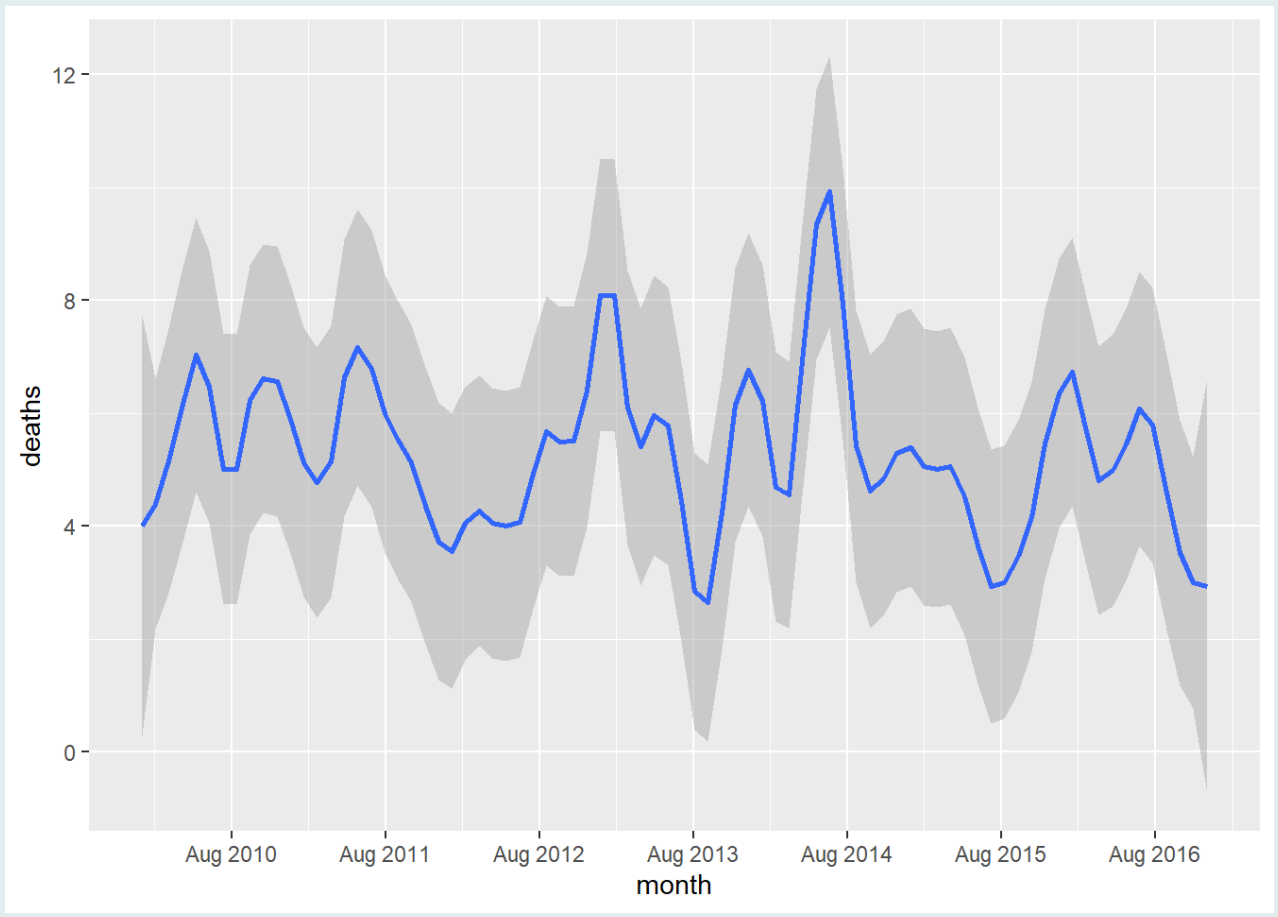
Q: Représentation des intervalles de confiance

```
nig_ken_mal %>%
  separate(malaria_cases,
           into = c("cases", "cases_lower", "cases_upper"),
           sep = "\\(|to") %>%
  mutate(across(c("cases", "cases_lower", "cases_upper"),
                ~ str_replace_all(.x, "[^0-9]", "") %>%
                  as.numeric()
                )) %>%
  ggplot(aes(x = year, y = cases, color = country, fill = country)) +
  geom_line() +
  geom_ribbon(aes(ymin = cases_lower, ymax = cases_upper), alpha = 0.4)
```



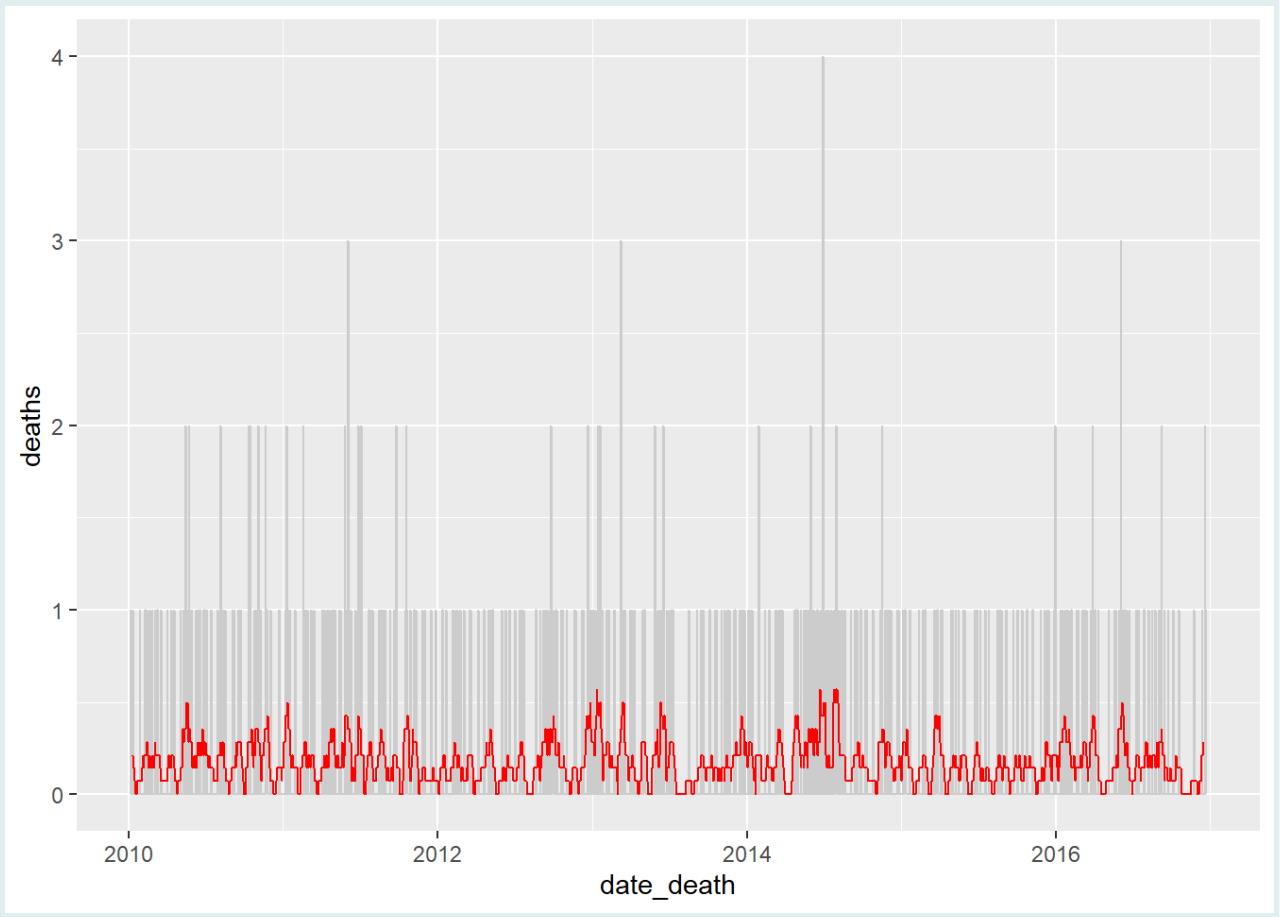
Q: Lissage des données sur les décès liés au VIH en Colombie

```
hiv_monthly_deaths_table <-
  colom_hiv_deaths %>%
  # Aggregate data to count deaths per month
  mutate(month = floor_date(date_death, unit = "month")) %>%
  group_by(month) %>%
  summarize(deaths = n())

# Create the epicurve
ggplot(hiv_monthly_deaths_table, aes(x = month, y = deaths)) +
  # Apply smoothing to the curve
  geom_smooth(method = "loess", span = 0.1) +
  scale_x_date(date_breaks = "12 months", date_labels = "%b %Y")
```



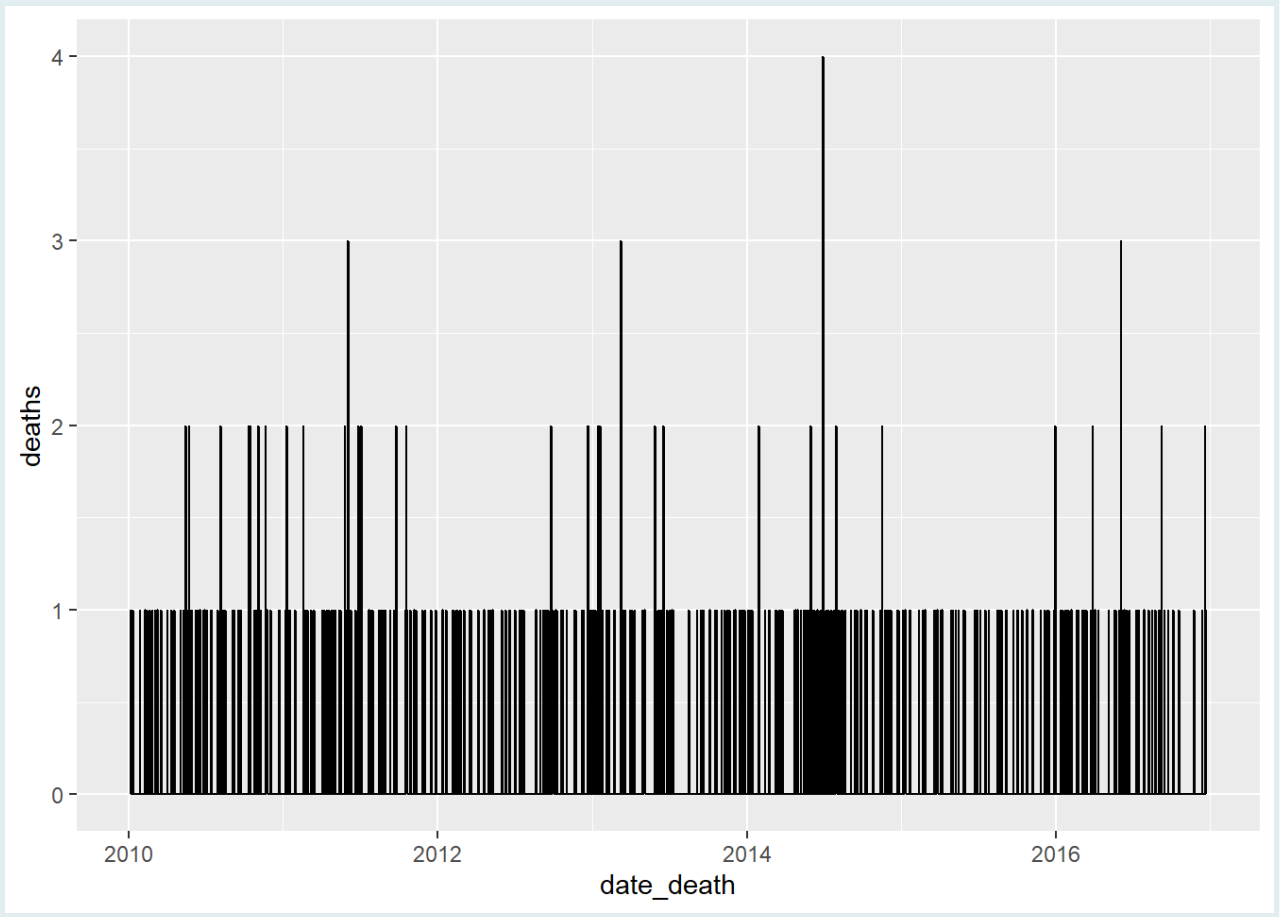
Q: Lissage avec des moyennes mobiles

```
colom_hiv_deaths_per_day %>%
  mutate(roll_deaths = rollmean(deaths, k = 14, fill = NA)) %>%
  ggplot(aes(x = date_death, y = deaths)) +
  geom_line(color = "gray80") +
  geom_line(aes(y = roll_deaths), color = "red")
```

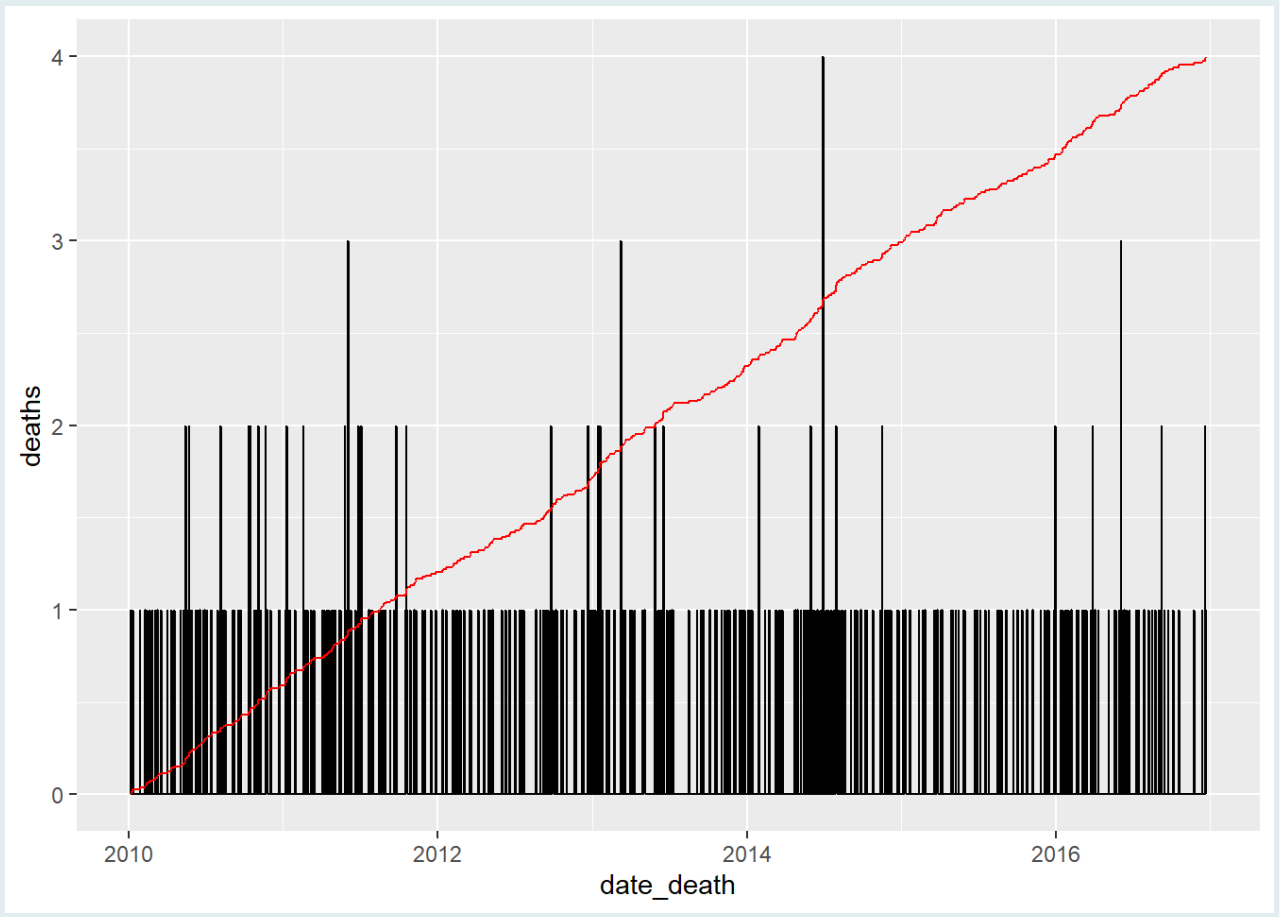


Q: Axes secondaires

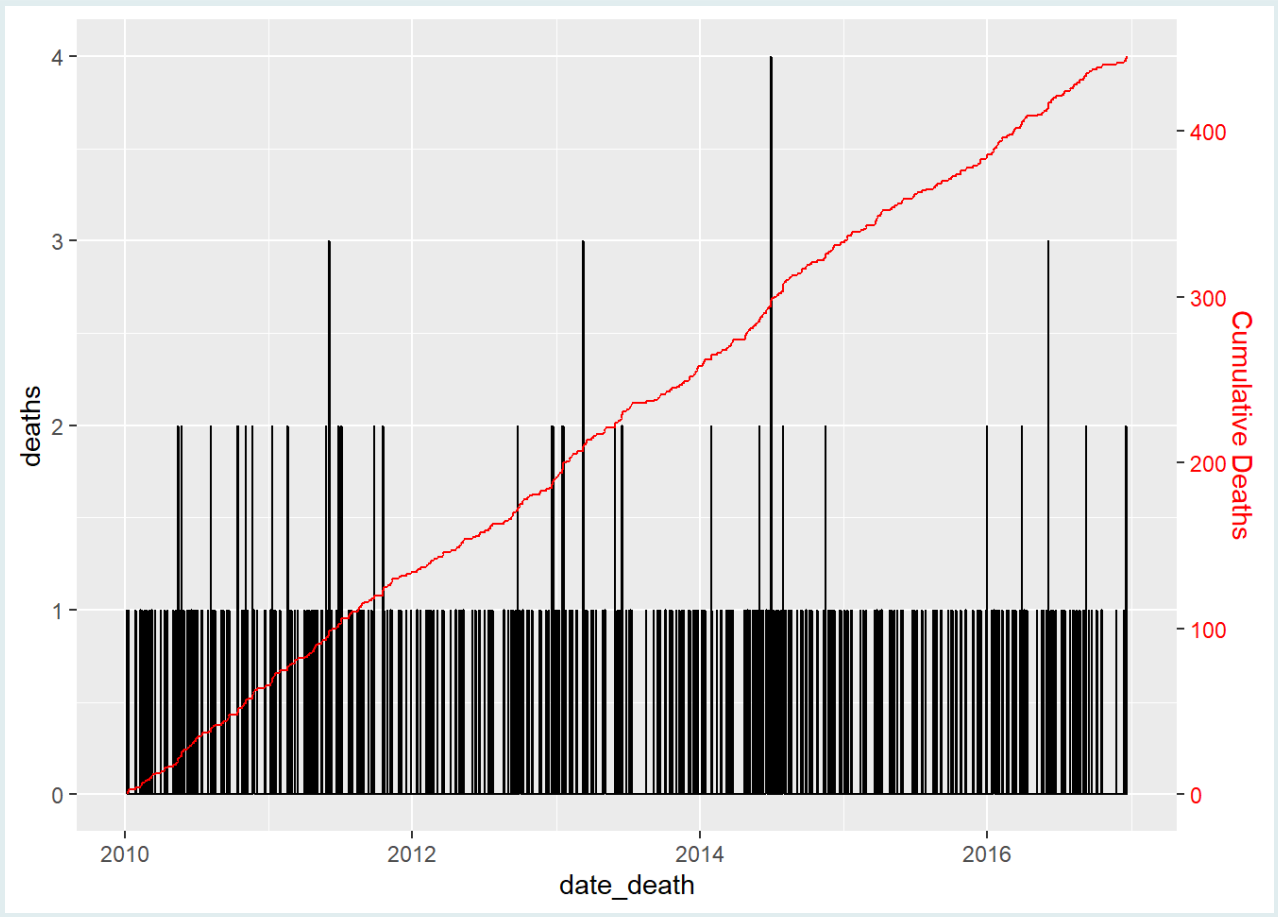
```r
# Step 1: Calculate cumulative deaths
colom_hiv_deaths_cumul <- colom_hiv_deaths_per_day %>%
  mutate(cum_deaths = cumsum(deaths))

# Step 2: Plot daily deaths
ggplot(colom_hiv_deaths_cumul, aes(x = date_death)) +
  geom_line(aes(y = deaths))
```




```r
# Step 3: Calculate scale factor
scale_factor <- max(colom_hiv_deaths_cumul$cum_deaths) /
        max(colom_hiv_deaths_cumul$deaths)

# Step 4: Add cumulative deaths to the plot
ggplot(colom_hiv_deaths_cumul, aes(x = date_death)) +
  geom_line(aes(y = deaths)) +
  geom_line(aes(y = cum_deaths / scale_factor), color = "red")
```




```
# Step 5: Add secondary y-axis
ggplot(colom_hiv_deaths_cumul, aes(x = date_death)) +
  geom_line(aes(y = deaths)) +
  geom_line(aes(y = cum_deaths / scale_factor), color = "red") +
  scale_y_continuous(sec.axis = sec_axis(trans = ~ .x * scale_factor, name =
        "Cumulative Deaths")) +
    theme(axis.text.y.right = element_text(color = "red"),
        axis.title.y.right = element_text(color = "red"))
```



Contributeurs

Les membres suivants de l'équipe 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

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement