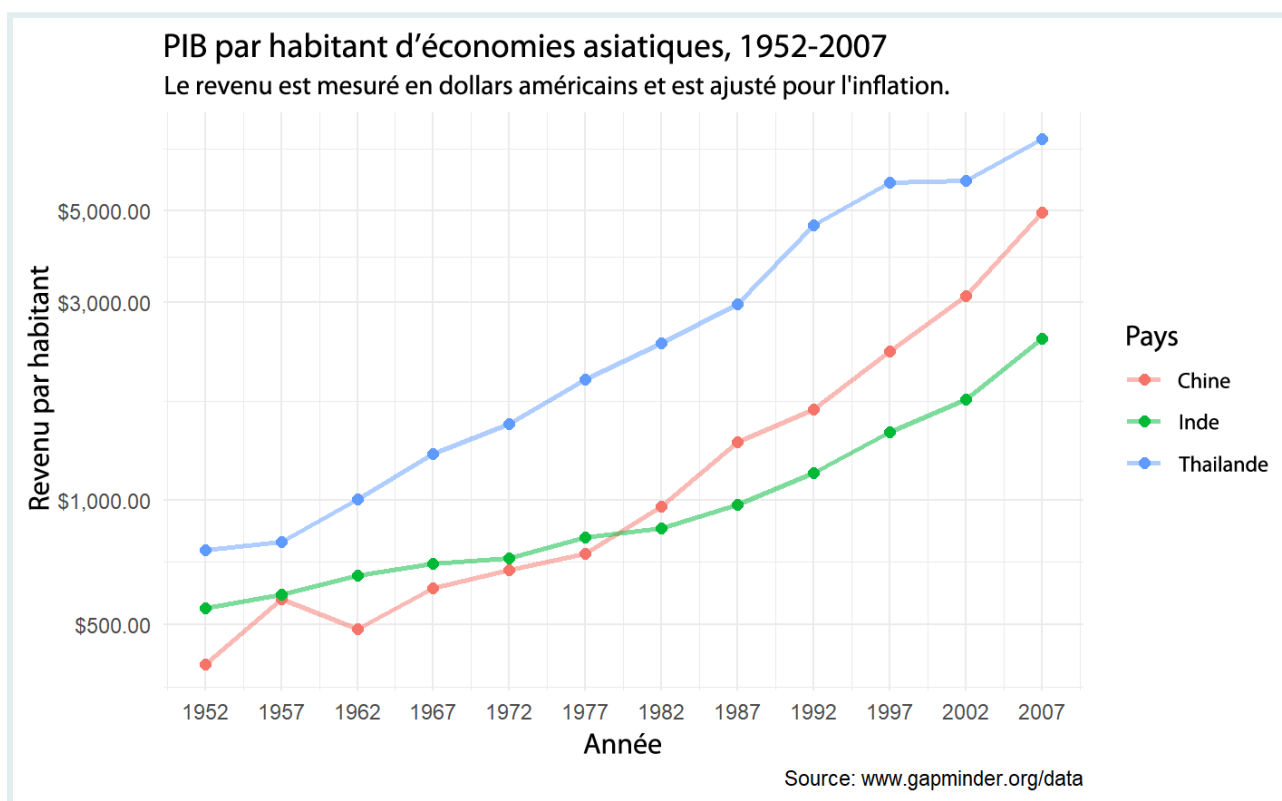

Lignes, échelles et étiquettes

March 2024

Objectifs d'Apprentissage
Introduction
Packages
Le dataframe <code>gapminder</code>
Graphiques linéaires avec <code>geom_line()</code>
Les esthétiques fixes dans <code>geom_line()</code>
Combiner les éléments géométriques
Mapper les données sur plusieurs lignes
Modifier les échelles continues x et y
Modifier les graduations des axes
Définir une échelle logarithmique
Étiquetage avec <code>labs()</code>
Preview : Les thèmes
En résumé
Contributeurs
Références

Objectifs d'Apprentissage

1. Créer des **graphiques linéaires** pour visualiser les relations entre deux variables numériques avec `geom_line()`.
2. **Ajouter des points** à un graphique linéaire avec `geom_point()`.
3. Utiliser des esthétiques comme `size`, `color`, et `linetype` pour modifier les graphiques linéaires.
4. **Manipuler les échelles (scale) des axes** de données continues avec `scale_*_continuous()` et `scale_*_log10()`.
5. **Ajouter des étiquettes (labels)** à un graphique tels que `title`, `subtitle`, ou `caption` avec la fonction `labs()`.



Introduction

Les graphiques linéaires sont utilisés pour montrer les **relations** entre deux **variables numériques**, tout comme les nuages de points. Ils sont particulièrement utiles lorsque la variable sur l'axe des x, également appelée variable *explicative*, est de nature **séquentielle**. En d'autres termes, il y a un ordre inhérent à la variable.

Les exemples les plus courants de graphiques linéaires incluent une composante **temporelle sur l'axe des x**, tels que les heures, les jours, les semaines ou les années. Étant donné que le temps est une séquence, nous relierons les observations

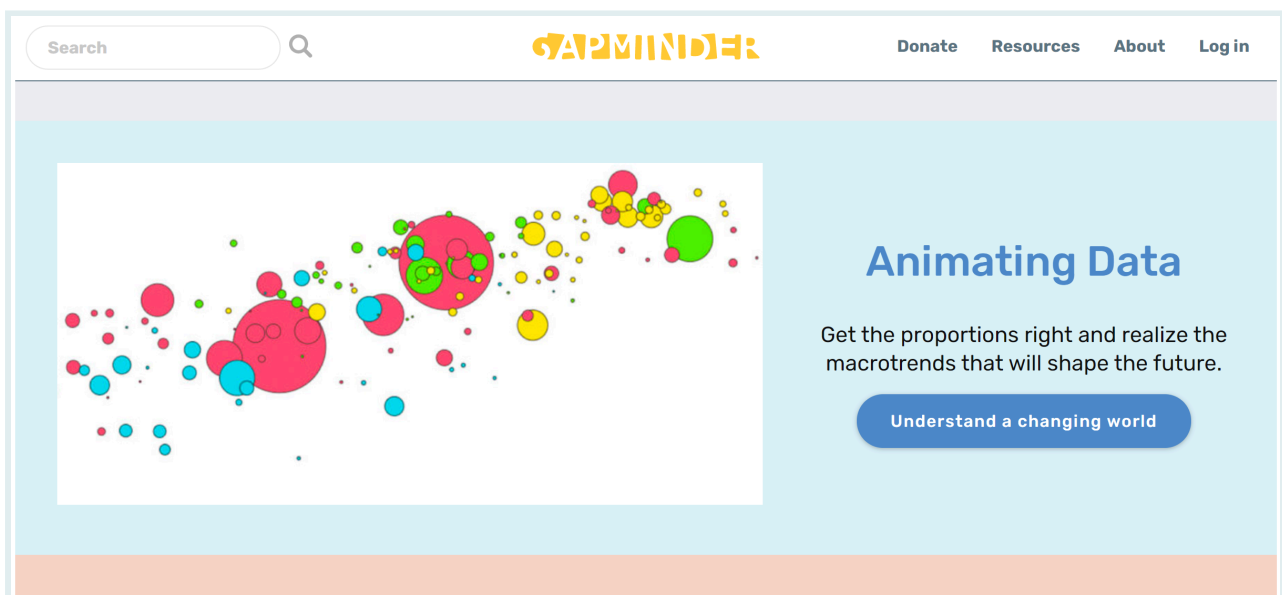
consécutives de la variable sur l'axe des y avec une ligne. Ces graphiques linéaires, qui intègrent une notion de temps sur l'axe des x, sont communément appelés des **graphiques de série temporelle**.

Packages

```
# Charger les packages
pacman::p_load(tidyverse,
               gapminder,
               here)
```

Le dataframe `gapminder`

En février 2006, un médecin suédois et défenseur des données nommé Hans Rosling a donné une célèbre conférence TED intitulée “**Les meilleures statistiques jamais vues**” où il a présenté des données économiques, sanitaires et de développement mondial compilées par la Fondation Gapminder.



Nous pouvons accéder à un subset de ces données avec le package R **{gapminder}**, que nous venons de charger.

```
# Charger le dataframe gapminder à partir du package gapminder
data(gapminder, package="gapminder")
```

```
# Afficher le dataframe gapminder
```

Chaque ligne de ce tableau correspond à une combinaison pays-année. Pour chaque ligne, nous avons 6 colonnes :

1. **country** : Nom du pays
2. **continent** : Région géographique du monde
3. **year** : Année calendaire
4. **lifeExp** : L'espérance de vie à la naissance en années
5. **pop** : Population totale
6. **gdpPercap** : Produit intérieur brut par personne (en dollar américain ajusté en fonction de l'inflation)

La fonction `str()` peut nous en apprendre plus sur ces variables.

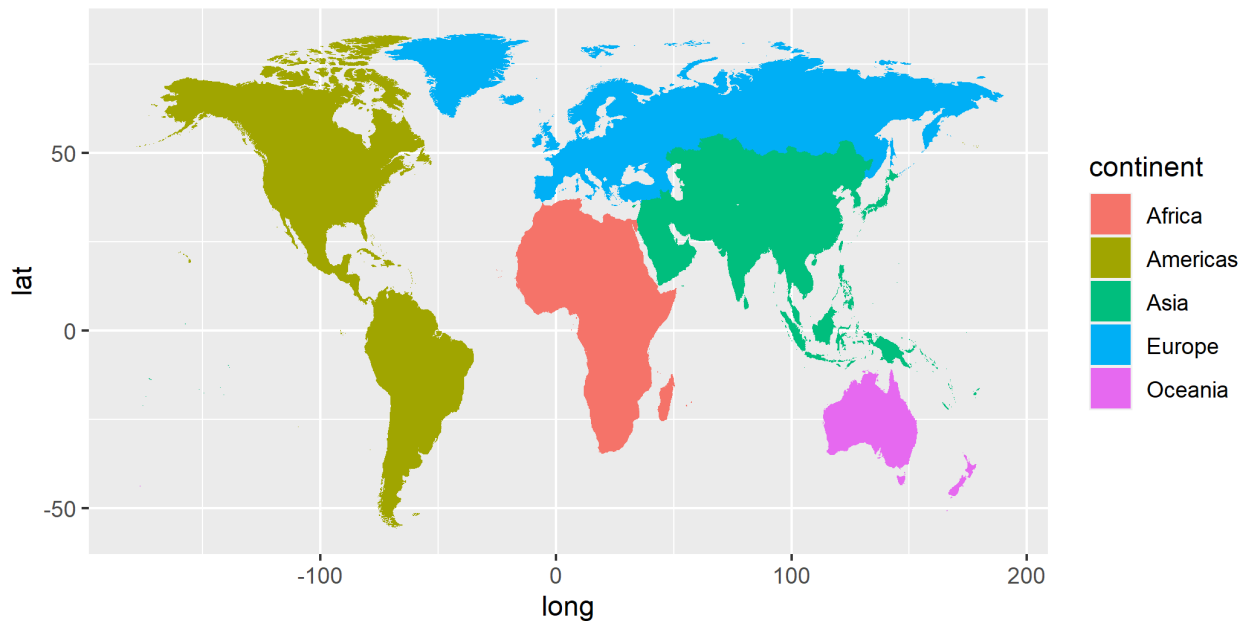
```
# Structure des données  
str(gapminder)
```

```
## tibble [1,704 × 6] (S3: tbl_df/tbl/data.frame)  
## $ country : Factor w/ 142 levels "Afghanistan",...: 1 1 1 1 1 1 1 1 1 1  
...  
## $ continent: Factor w/ 5 levels "Africa","Americas",...: 3 3 3 3 3 3 3 3 3 3  
3 ...  
## $ year      : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992  
1997 ...  
## $ lifeExp   : num [1:1704] 28.8 30.3 32 34 36.1 ...  
## $ pop       : int [1:1704] 8425333 9240934 10267083 11537966 13079460  
14880372 12881816 13867957 16317921 22227415 ...  
## $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

Cette version du dataset `gapminder` contient des informations sur **142 pays** répartis en **5 continents**.

Gapminder world regions

Five regions in the `continent` variable of `gapminder`



```
# Résumé des données  
summary(gapminder)
```

```
##           country           continent  
## Afghanistan: 12 Africa :624  
## Albania      : 12 Americas:300  
## Algeria      : 12 Asia :396  
## Angola       : 12 Europe :360  
## Argentina    : 12 Oceania : 24  
## Australia    : 12  
## (Other)      :1632  
##           year           lifeExp  
## Min.       :1952 Min.       :23.60  
## 1st Qu.:1966 1st Qu.:48.20  
## Median :1980 Median :60.71  
## Mean      :1980 Mean      :59.47  
## 3rd Qu.:1993 3rd Qu.:70.85  
## Max.      :2007 Max.      :82.60  
##  
##           pop           gdpPercap  
## Min.      :6.001e+04 Min.      : 241.2  
## 1st Qu.:2.794e+06 1st Qu.: 1202.1  
## Median :7.024e+06 Median : 3531.8  
## Mean      :2.960e+07 Mean      : 7215.3  
## 3rd Qu.:1.959e+07 3rd Qu.: 9325.5  
## Max.      :1.319e+09 Max.      :113523.1  
##
```

Les données sont enregistrées tous les 5 ans de 1952 à 2007 (soit un total de 12 années).

Supposons que nous voulions visualiser la relation entre le temps (`year`) et l'espérance de vie (`lifeExp`).

Pour l'instant, concentrons-nous uniquement sur un pays - les États-Unis. D'abord, nous devons créer un nouveau dataframe avec uniquement les données de ce pays.

```
# Sélectionner les cas US
gap_US <- dplyr::filter(gapminder,
                        country == "United States")

gap_US
```

REMINDER



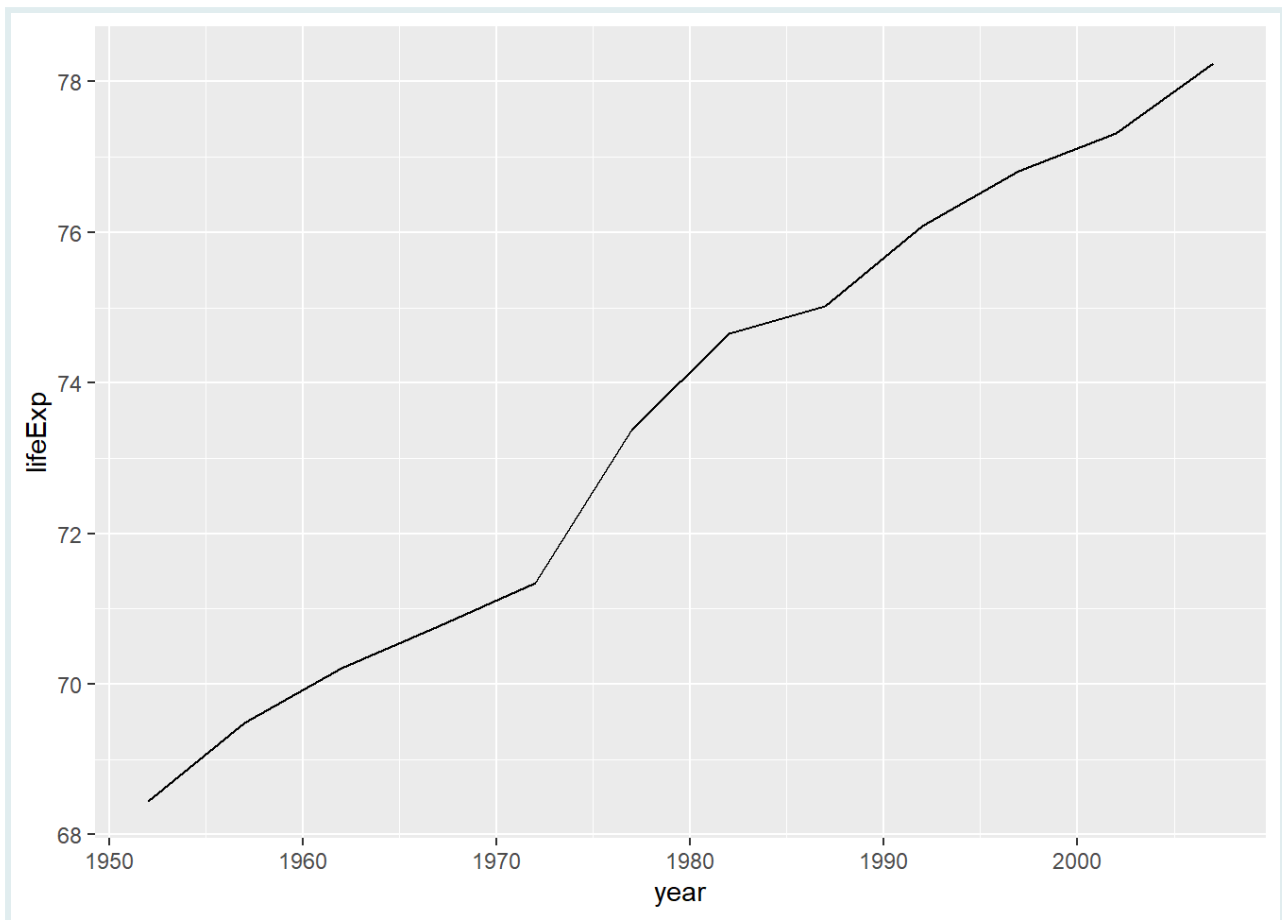
Le code ci-dessus est couvert dans notre cours sur la manipulation de données en utilisant le package `{dplyr}`. La manipulation de données est le processus de transformation et de nettoyage des données dans le but de les rendre plus appropriées à des fins d'analyse. Par exemple, ce code utilise la fonction `filter()` pour créer un nouveau dataframe (`gap_US`) en incluant uniquement les lignes du dataframe `gapminder` qui ont "United States" dans la colonne `country`.

Graphiques linéaires avec `geom_line()`

Nous allons utiliser le dataframe `gap_US` avec `ggplot()` pour tracer **le temps** en années sur l'axe des abscisses `x` et **l'espérance de vie** sur l'axe des ordonnées `y`.

Nous pouvons visualiser les données de séries temporelles en utilisant `geom_line()` pour créer un graphique linéaire, au lieu d'utiliser `geom_point()` comme nous l'avons fait précédemment pour créer un nuage de points :

```
# Graphique linéaire simple
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line()
```

Tout comme avec le code `ggplot()` qui a créé le nuage de points de l'âge et de la charge virale avec `geom_point()`, décomposons ce code couche par couche en utilisant la grammaire des graphiques :

Dans l'appel de la fonction `ggplot()`, nous précisons deux des composants de la grammaire des graphiques comme arguments :

1. Le dataframe `gap_US` comme couche de données en réglant `data = gap_US`.
2. Le mapping esthétique `aes` en réglant `mapping = aes(x = year, y = lifeExp)`. Plus précisément, la variable `year` est associée à l'esthétique de position `x`, tandis que la variable `lifeExp` est associée à l'esthétique de position `y`.

Après avoir précisé à R quelles données et quelles correspondances esthétiques nous voulions tracer, nous allons ajouté le troisième composant essentiel, l'objet géométrique en utilisant l'opérateur `+`. Dans ce cas, l'objet géométrique a été réglé sur des lignes en utilisant `geom_line()`.

PRACTICE



(in RMD)

PRACTICE



Créez un graphique de série temporelle du PIB par habitant (`gdpPercap`) à partir du dataframe `gap_US` en utilisant `geom_line()` pour créer un graphique linéaire.

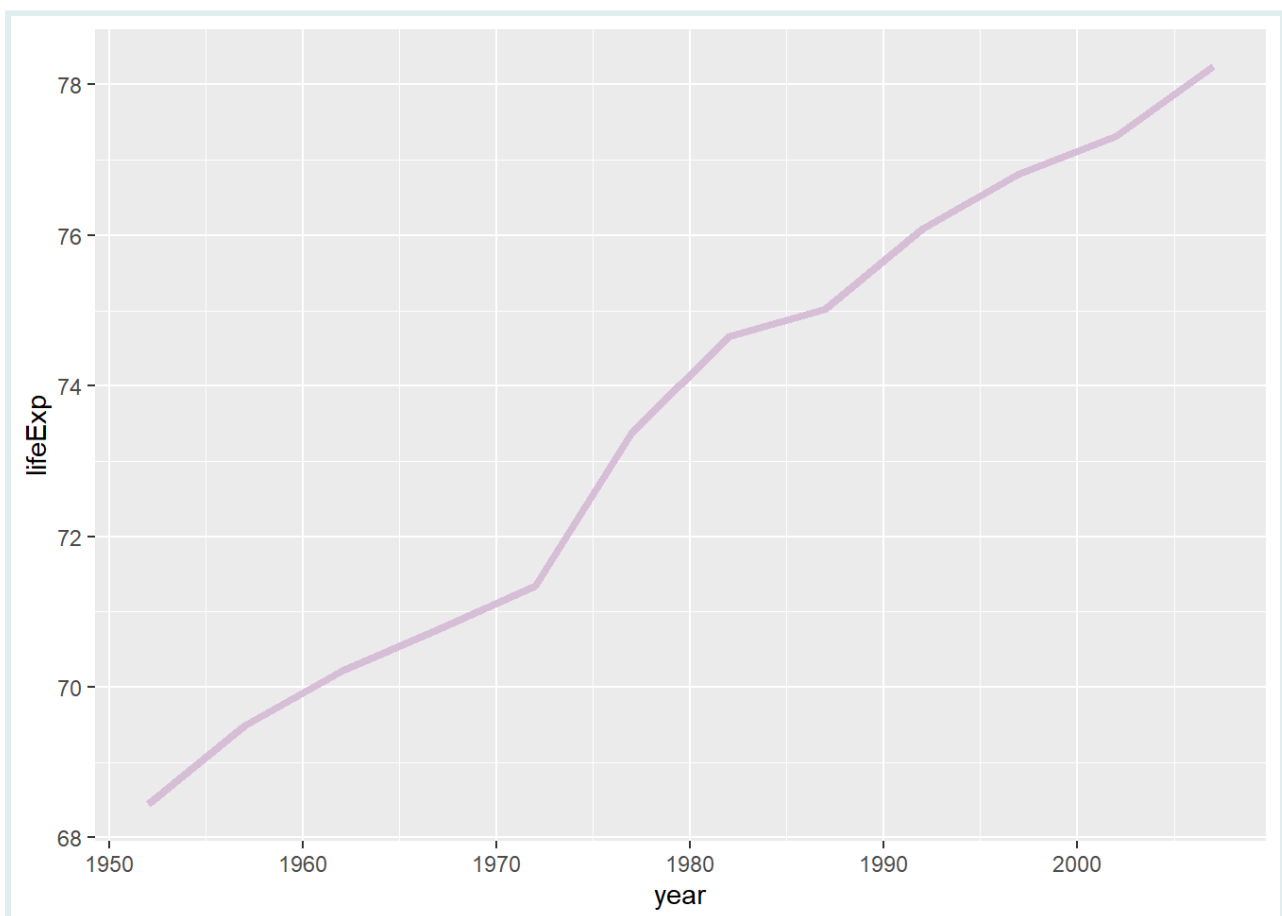
Les esthétiques fixes dans `geom_line()`

La couleur, l'épaisseur des lignes et le type de ligne du graphique linéaire peuvent être personnalisés en utilisant les arguments `color`, `size` et `linetype`, respectivement.


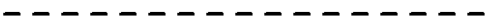




Nous avons changé la couleur et la taille des géométries dans le cours précédent.

Nous allons les réutiliser comme esthétiques fixes :

```
# Améliorer le graphique linéaire en ajoutant la couleur et la taille comme
# esthétiques fixes
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(color = "thistle",
           size = 1.5)
```

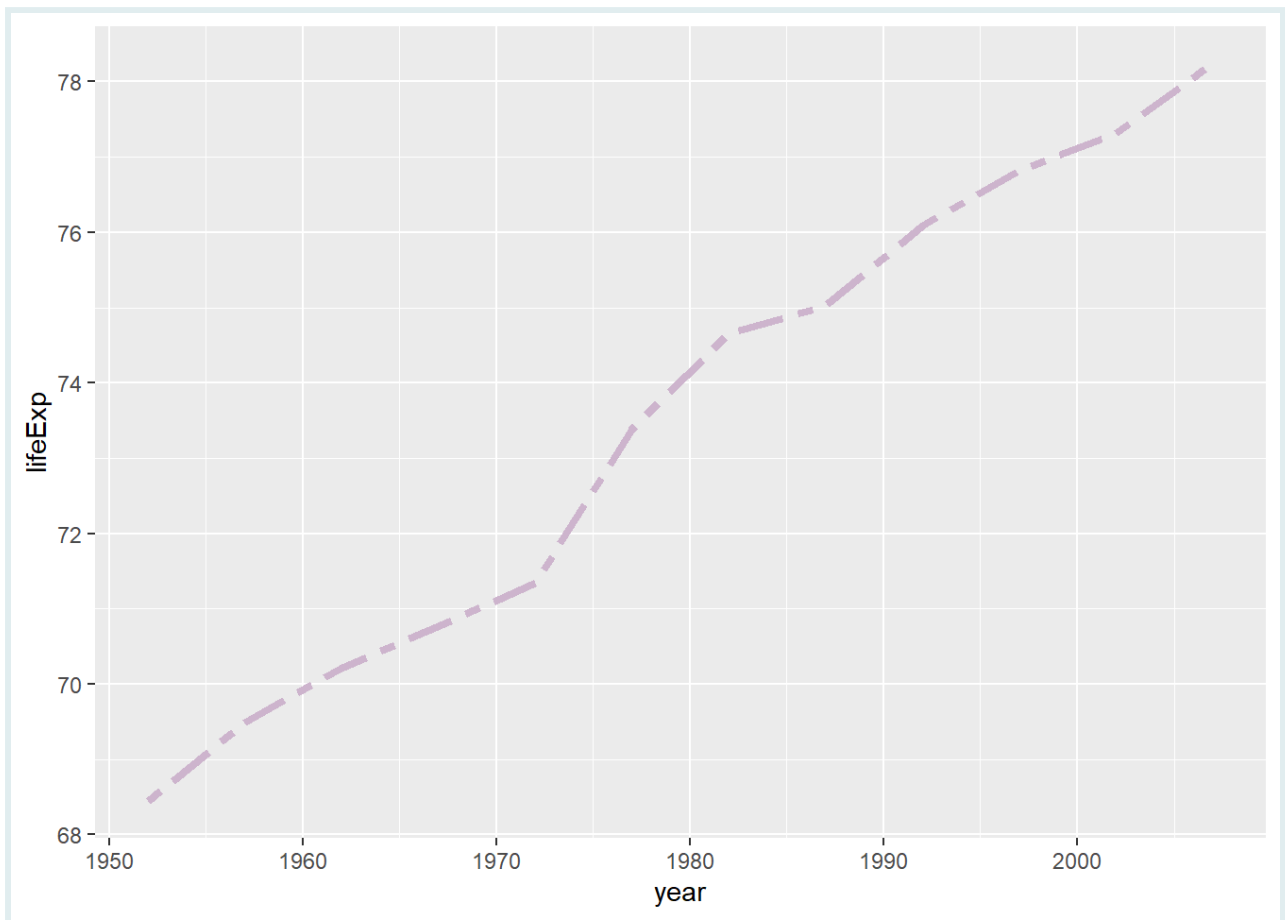


Nous allons maintenant introduire une nouvelle esthétique fixe qui est spécifique aux graphiques linéaires : `linetype` (ou `lty` en abrégé).

	<code>lty = 0</code> or 'blank'
	<code>lty = 1</code> or 'solid'
	<code>lty = 2</code> or 'dashed'
	<code>lty = 3</code> or 'dotted'
	<code>lty = 4</code> or 'dotdash'
	<code>lty = 5</code> or 'longdash'
	<code>lty = 6</code> or 'twodash'

Le type de ligne dans un graphique peut être spécifié en utilisant un nom ou un entier. Les types de ligne valides peuvent être définis à l'aide de chaînes de caractères compréhensibles : "blank", "solid", "dashed", "dotted", "dotdash", "longdash", et "twodash" sont tous compris par `linetype` ou `lty`.

```
# Améliorer le graphique linéaire en ajoutant la couleur, l'épaisseur et le
  type de ligne comme esthétiques fixes
ggplot(data = gap_US,
  mapping = aes(x = year,
                 y = lifeExp)) +
  geom_line(color = "thistle3",
            size = 1.5,
            linetype = "twodash")
```



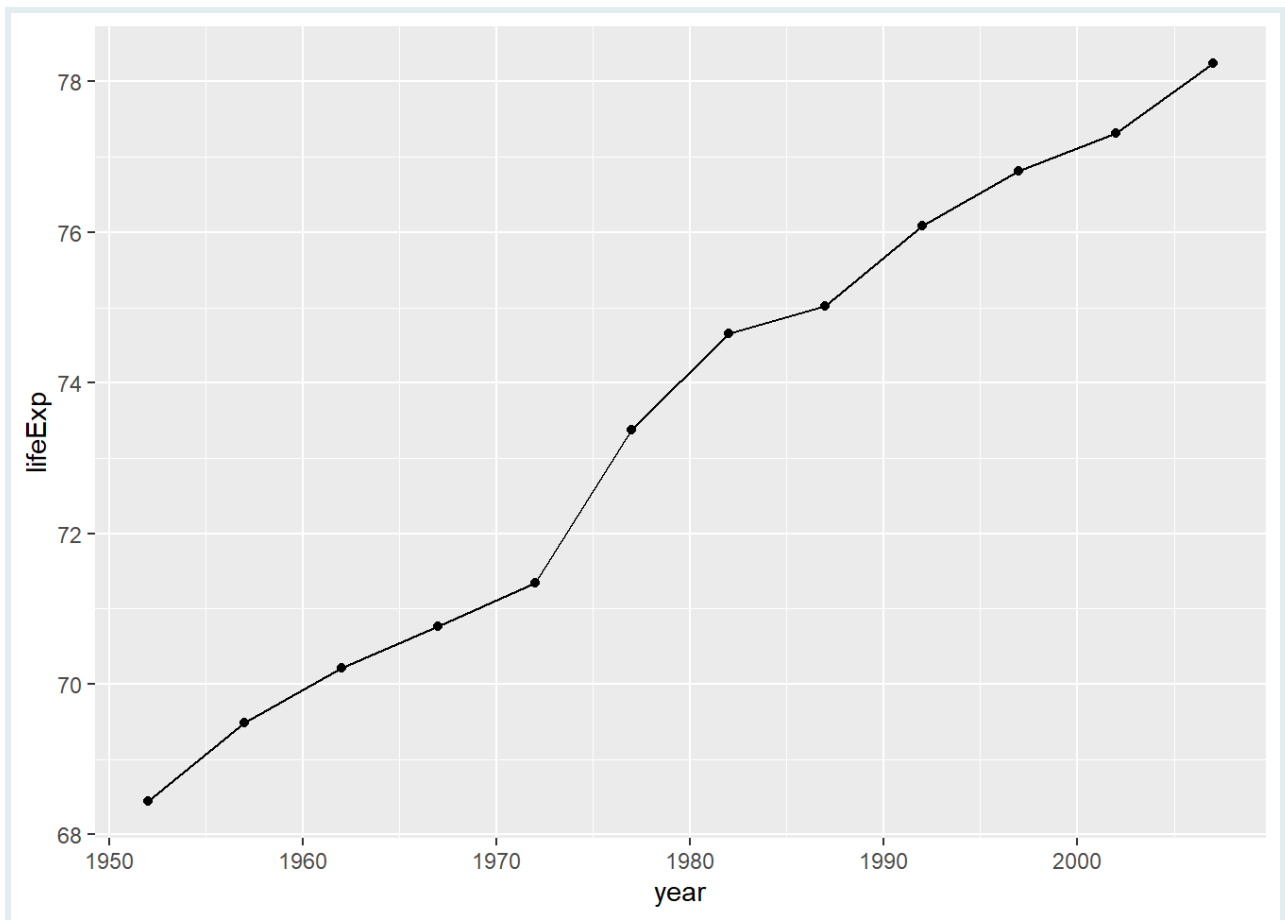
Dans les graphiques linéaires, il peut parfois être difficile de déterminer l'emplacement exact des points de données. Dans le graphique suivant, nous allons inclure des points pour une meilleure visualisation.

Combiner les éléments géométriques

Tant que les géométries sont compatibles, nous pouvons les superposer les unes sur les autres pour personnaliser davantage un graphique.

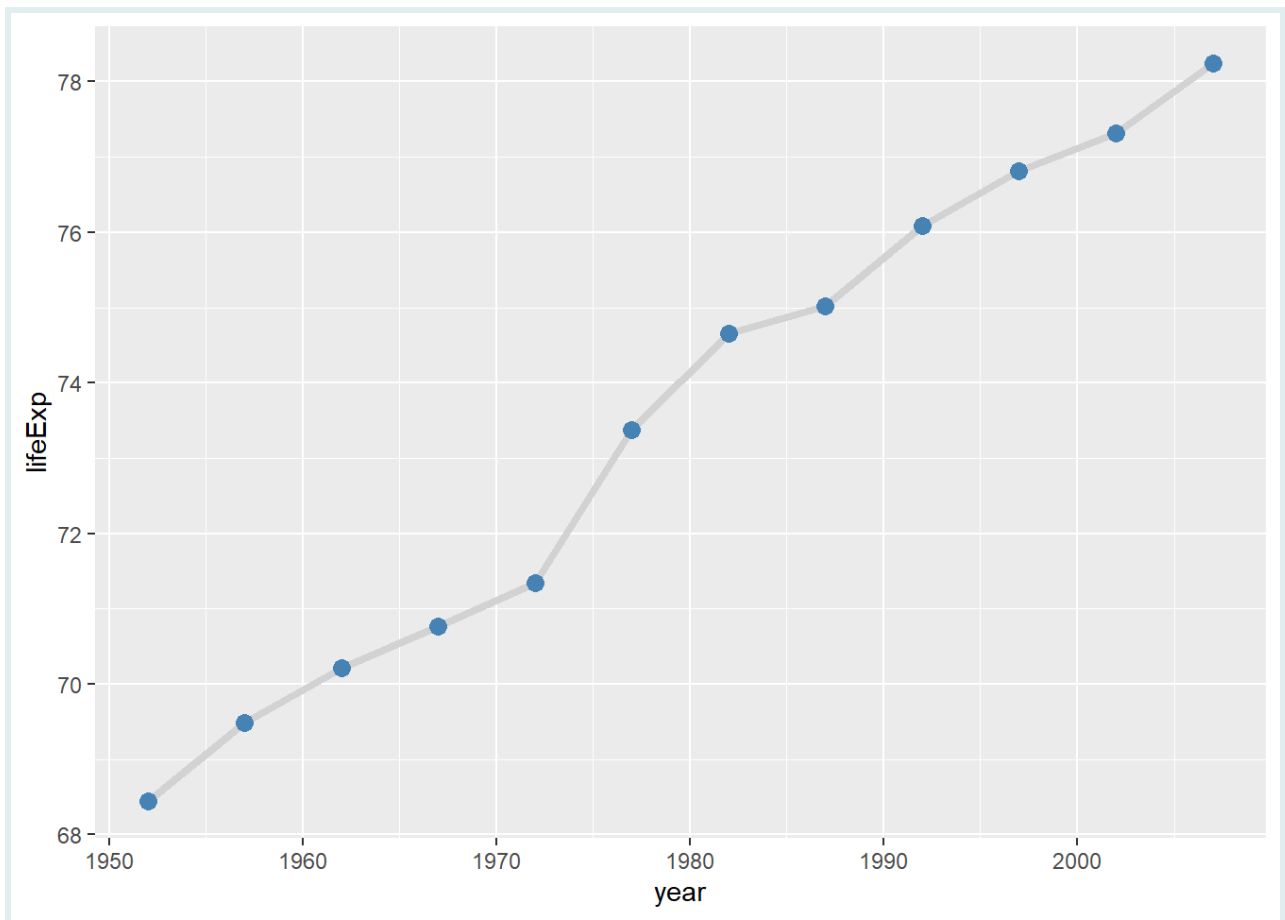
Par exemple, nous pouvons ajouter des points à notre graphique linéaire en utilisant l'opérateur `+` pour ajouter une seconde couche de `geom` avec `geom_point()` :

```
# Graphique linéaire simple avec des points
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line() +
  geom_point()
```



Nous pouvons améliorer l'apparence du graphique en personnalisant la taille et la couleur de nos géométries.

```
# Graphique linéaire avec des points et des esthétiques fixes
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(size = 1.5,
           color = "lightgrey") +
  geom_point(size = 3,
            color = "steelblue")
```



PRACTICE



(in RMD)

En vous basant sur le code ci-dessus, visualisez la relation entre le temps et le **PIB par habitant** (`gdpPerCap`) à partir du dataframe `gap_US`.

Utilisez à la fois des points et des lignes pour représenter les données.

Changez le type de ligne et la couleur des points par n'importe quelle valeur valide de votre choix.

Mapper les données sur plusieurs lignes

Dans la section précédente, nous n'avons examiné les données que d'un seul pays, mais que faire si nous voulons tracer les données de plusieurs pays et les comparer ?

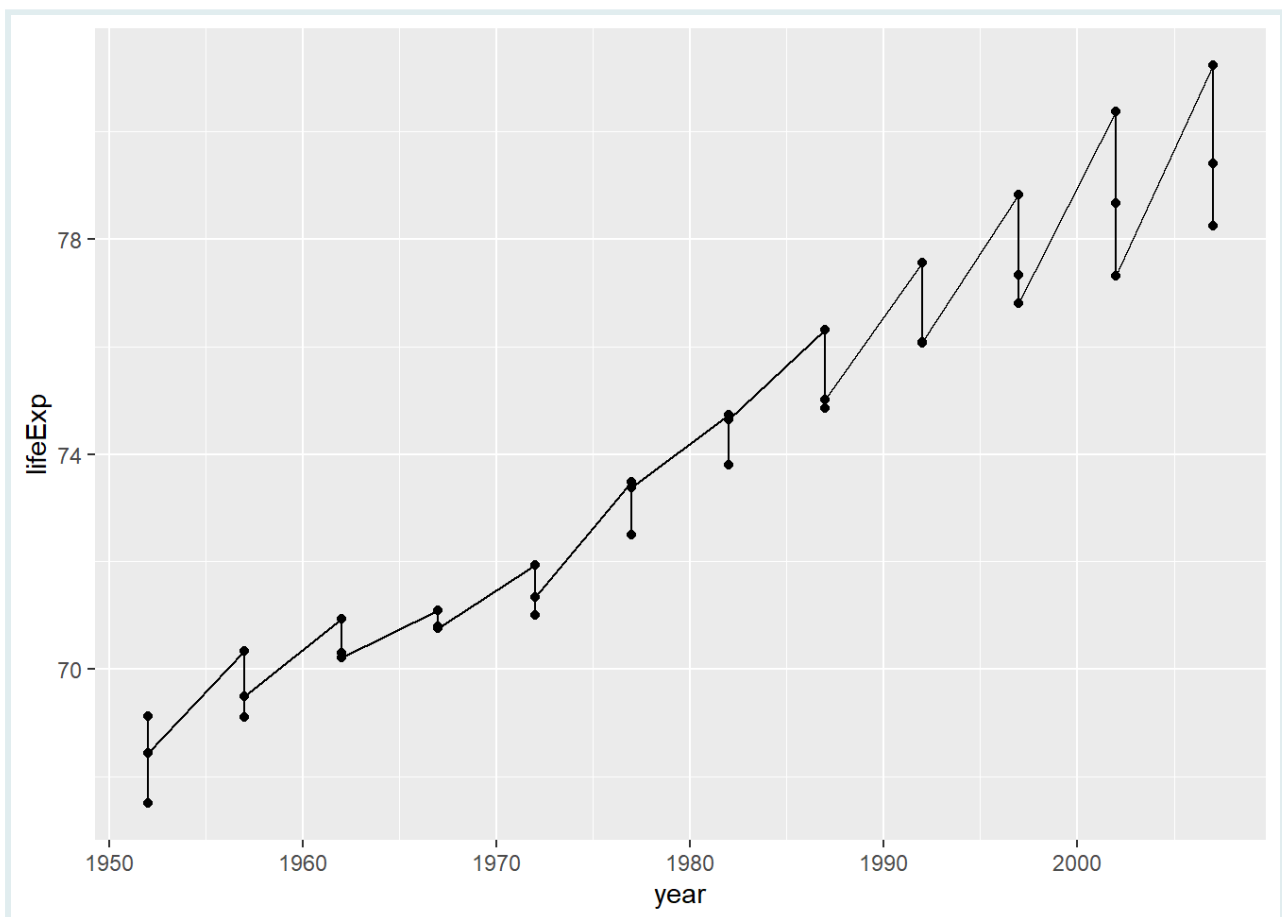
D'abord, ajoutons deux autres pays à notre subset :

```
# Créer un subset pour visualiser plusieurs catégories
gap_mini <- filter(gapminder,
                   country %in% c("United States",
                                   "Australia",
                                   "Germany"))

gap_mini
```

Lorsque nous remplaçons `gap_US` par `gap_mini` dans notre code, les lignes ne sont pas automatiquement séparées par pays :

```
# Graphique en ligne sans esthétique de groupe
ggplot(data = gap_mini,
       mapping = aes(y = lifeExp,
                     x = year)) +
  geom_line() +
  geom_point()
```

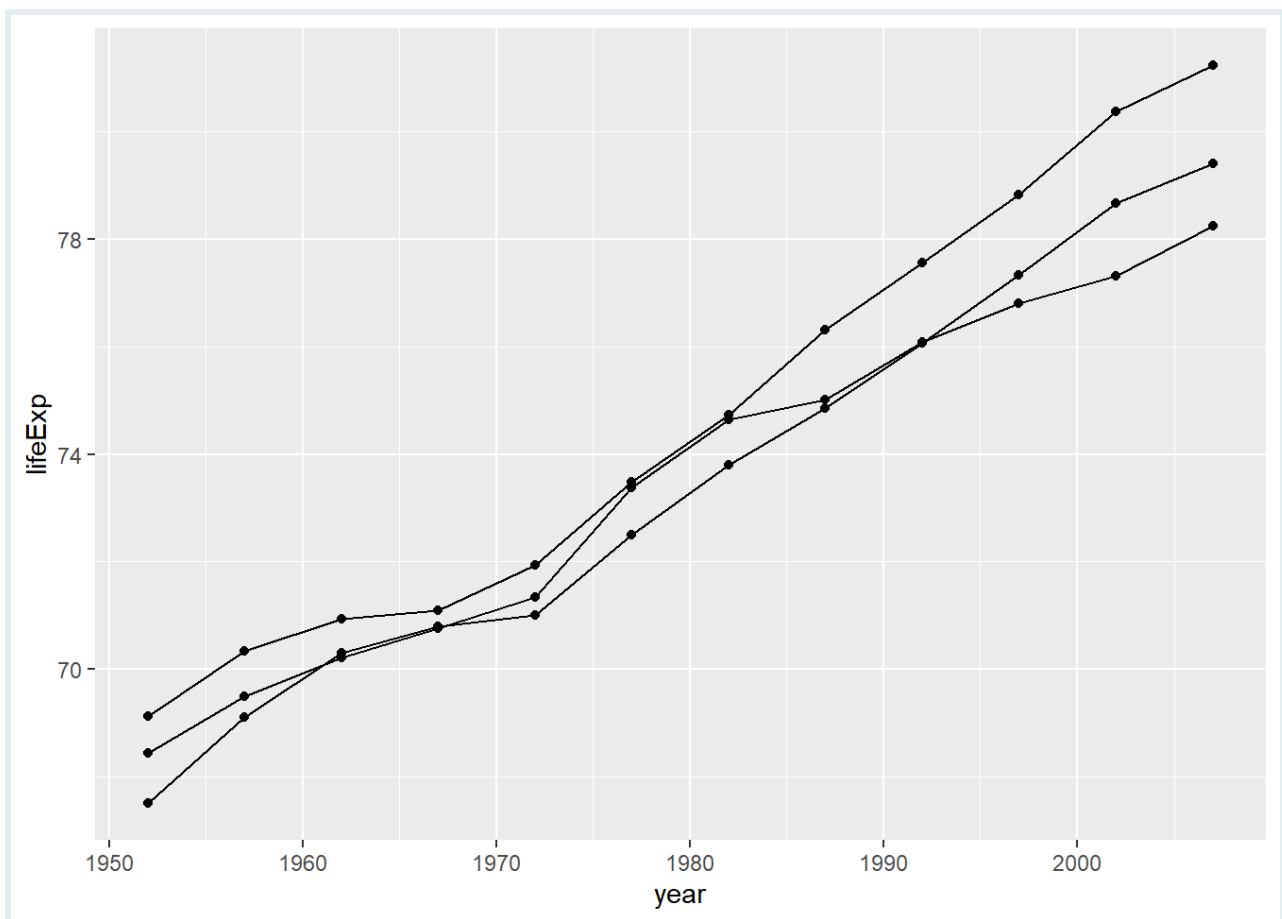


Ce graphique n'est pas très utile pour faire des comparaisons entre les groupes.

Pour indiquer à `ggplot()` de mapper les données de chaque pays séparément, nous pouvons utiliser l'argument `group` comme mapping esthétique :

```
# Graphique linéaire avec regroupement par une variable catégorielle
ggplot(data = gap_mini,
       mapping = aes(y = lifeExp,
                     x = year,
                     group = country)) +

geom_line() +
geom_point()
```

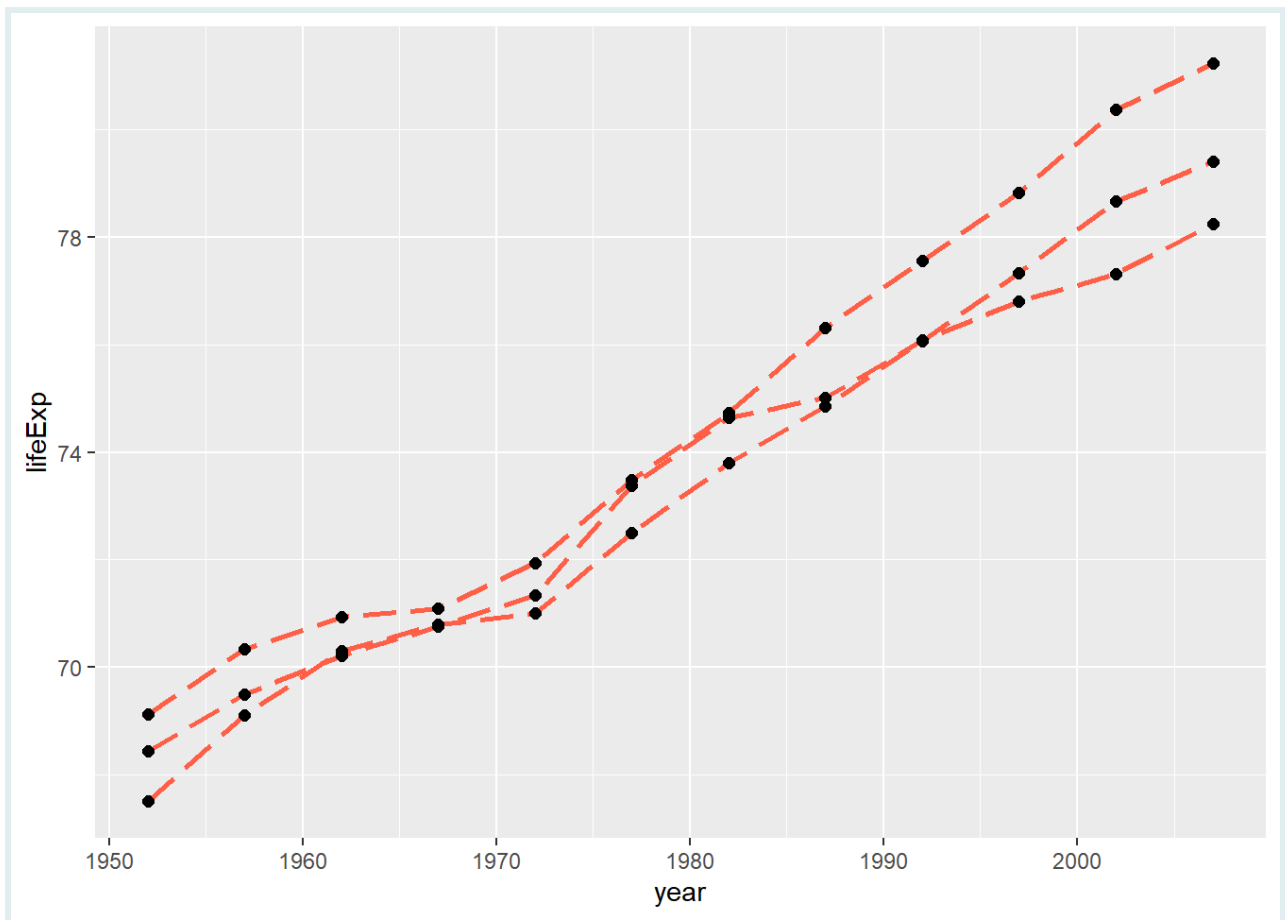


Maintenant que les données sont groupées par pays, nous avons 3 lignes séparées - une pour chaque modalité de la variable `country`.

Nous pouvons également appliquer des esthétiques fixes aux couches géométriques.

```
# Appliquer des esthétiques fixes à plusieurs lignes
ggplot(data = gap_mini,
       mapping = aes(y = lifeExp,
                     x = year,
                     group = country)) +

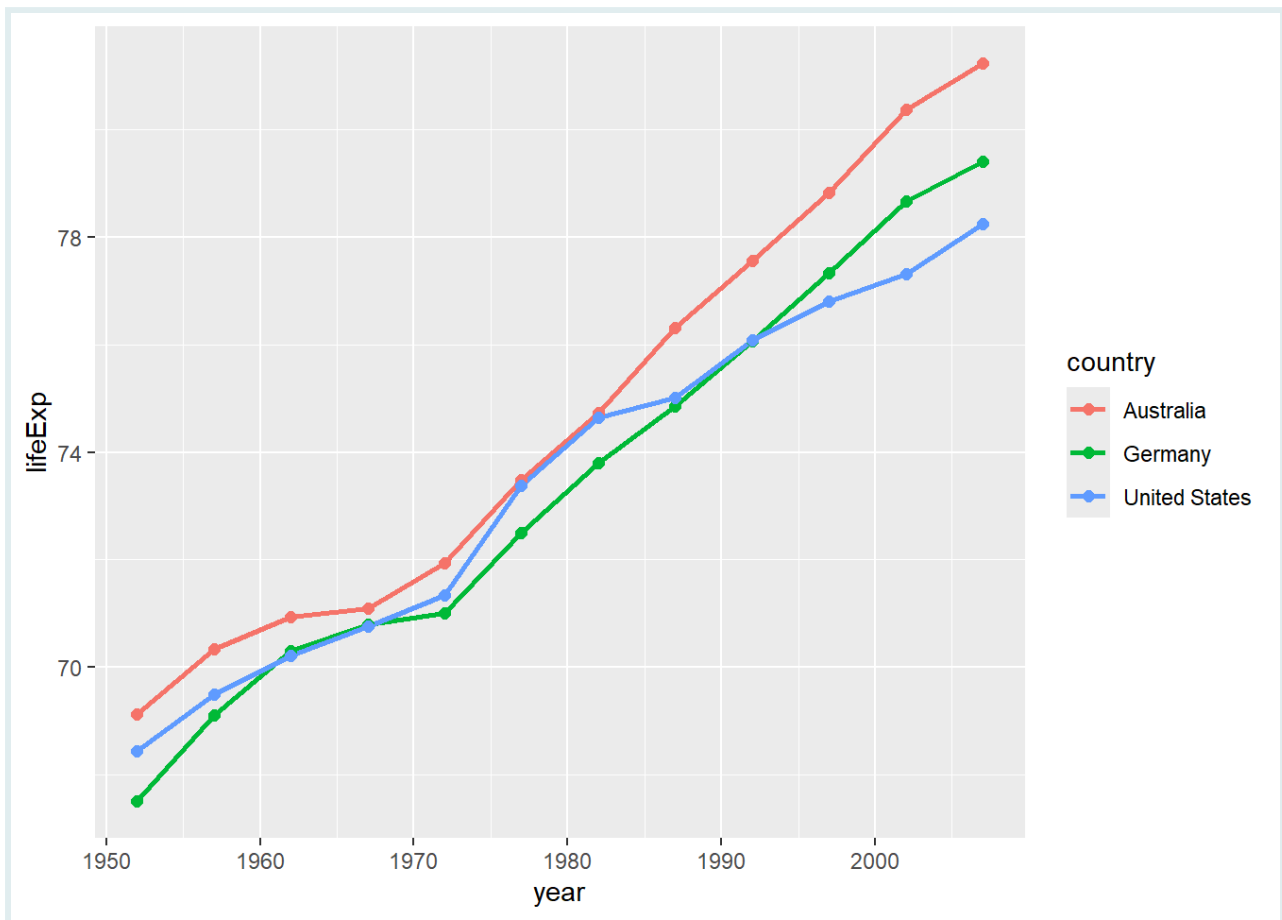
geom_line(linetype="longdash",      # définir le type de ligne
          color="tomato",           # définir la couleur de la ligne
          size=1) +                 # définir l'épaisseur de la ligne
geom_point(size = 2)               # définir la taille du point
```

Dans le graphiques ci-dessus, le type, la couleur et l'épaisseur des lignes sont les mêmes pour les trois groupes.

Cela ne nous permet pas de distinguer les groupes. Il faut ajouter un mapping esthétique qui peut nous aider à identifier à quel pays appartient chaque ligne, comme la couleur ou le type de ligne.

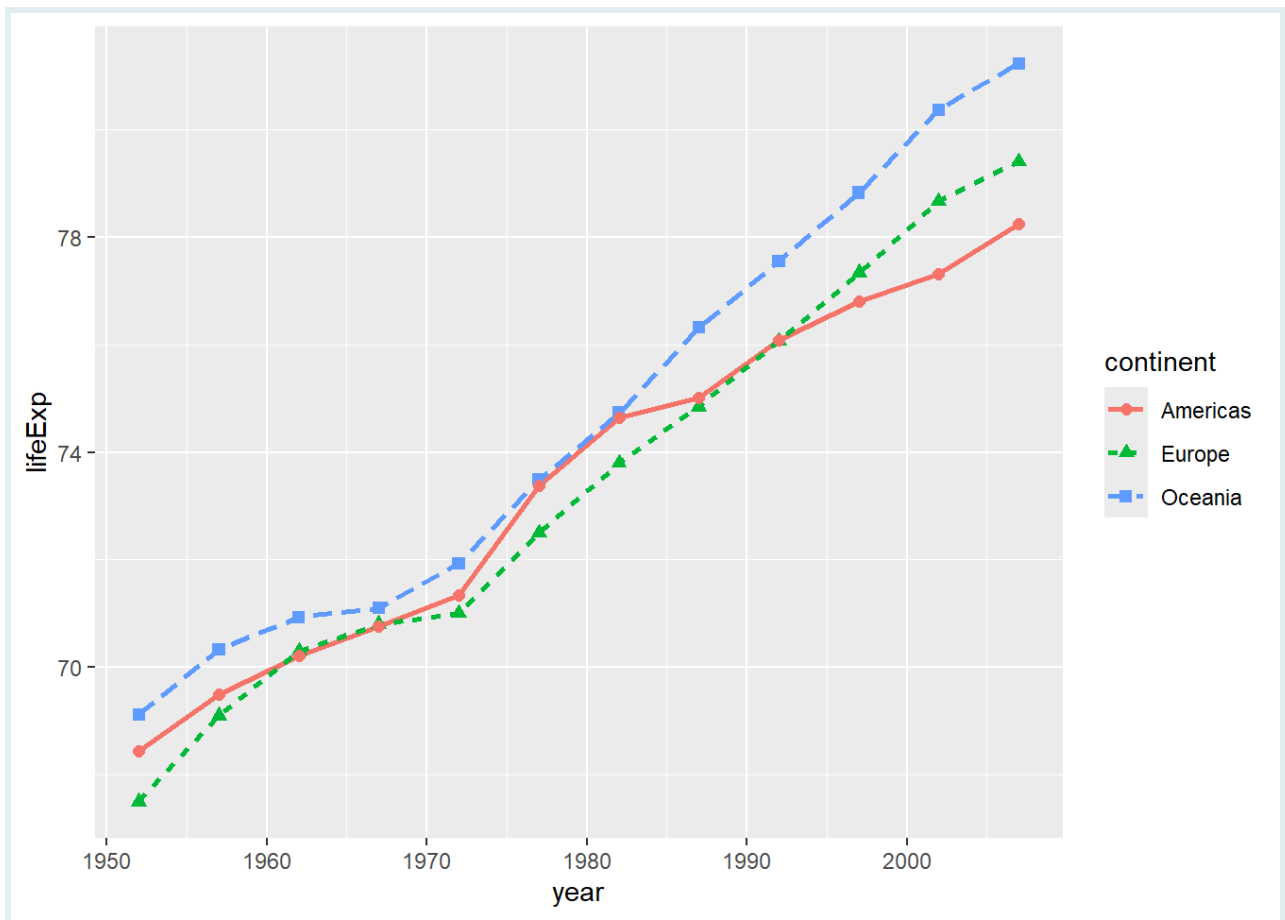
```
# Mapper le pays à la couleur
ggplot(data = gap_mini,
       mapping = aes(y = lifeExp, x = year,
                     group = country,
                     color = country)) +
  geom_line(size = 1) +
  geom_point(size = 2)
```



Les mappings esthétiques spécifiés dans l'appel de la fonction `ggplot()` sont transmis aux couches ultérieures.

Au lieu de grouper par `country`, nous pouvons également grouper par `continent` :

```
# Mapper le continent à la couleur, au type de ligne et à la forme des poits
ggplot(data = gap_mini,
       mapping = aes(x = year,
                     y = lifeExp,
                     color = continent,
                     lty = continent,
                     shape = continent)) +
  geom_line(size = 1) +
  geom_point(size = 2)
```



Lorsque nous fournissons plusieurs mappings et geoms, `{ggplot2}` peut discerner quels mappings s'appliquent à quels geoms.

Ici, `color` a été appliqué aux points et aux lignes, mais `lty` a été ignoré par `geom_point()` et `shape` a été ignoré par `geom_line()`, puisqu'ils ne peuvent pas être appliqués.

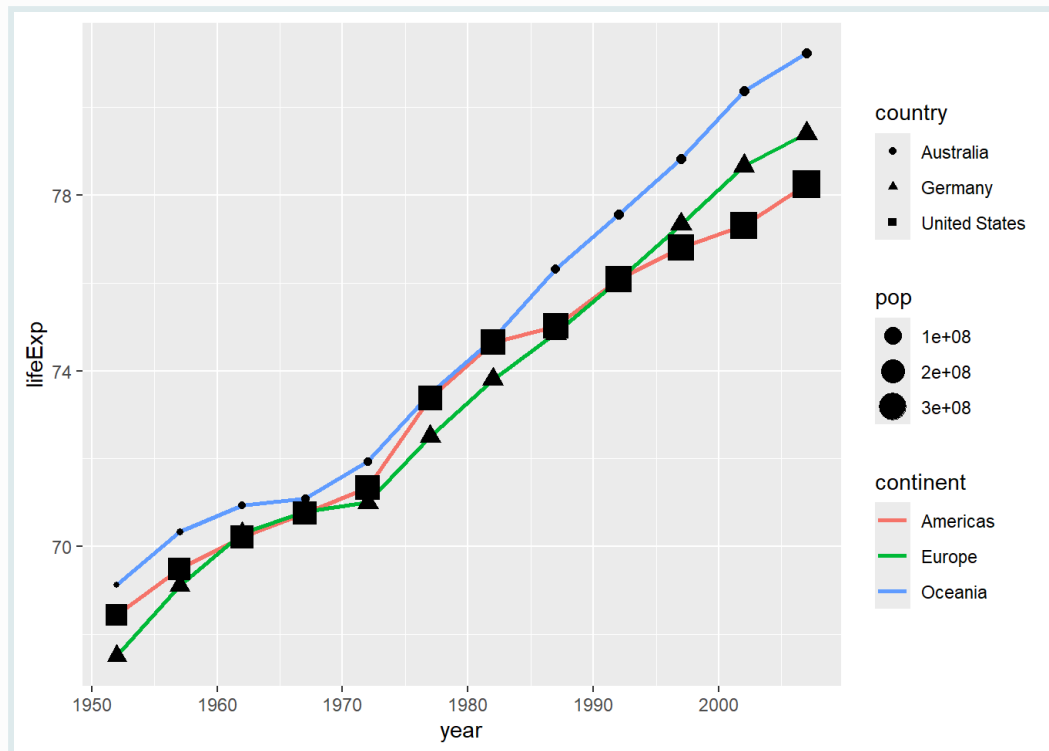
Les mappings sont inclus soit dans la fonction `ggplot()`, soit dans la couche `geom_*()`.



Par exemple, les mappings esthétiques peuvent aller dans `geom_line()` et ne seront appliqués qu'à cette couche :

```
ggplot(data = gap_mini,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(size = 1, mapping = aes(color = continent)) +
  geom_point(mapping = aes(shape = country,
                           size = pop))
```

CHALLENGE

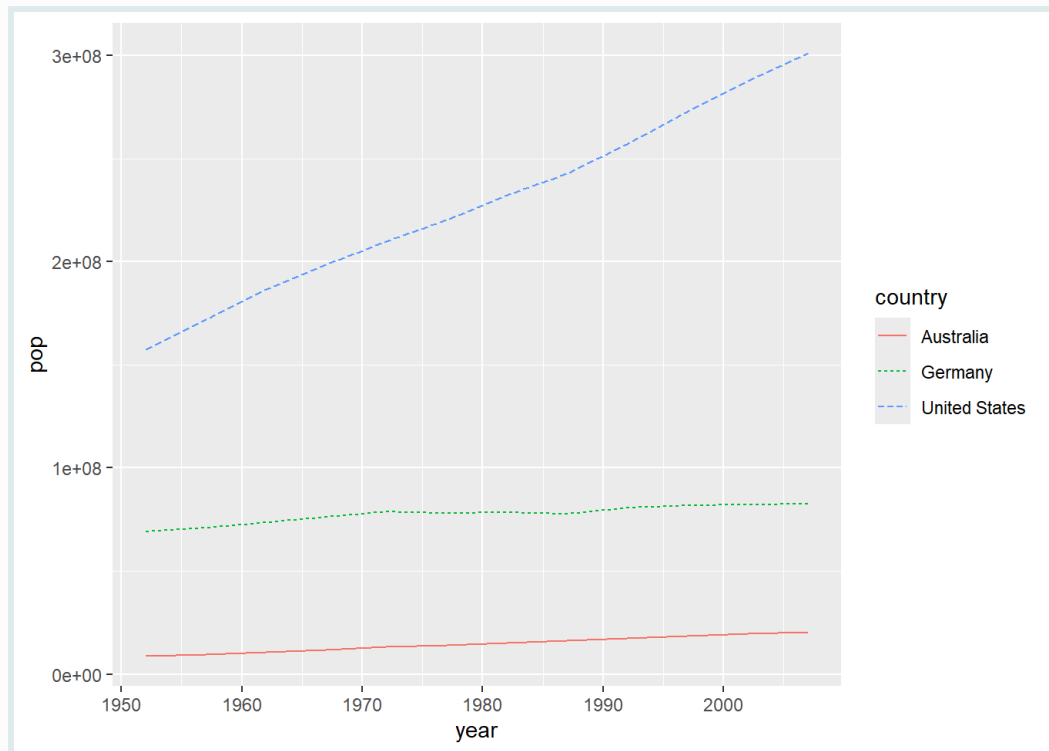


Essayez d'ajouter `mapping= aes()` dans `geom_point()` et mappez `continent` à une esthétique appropriée !

PRACTICE



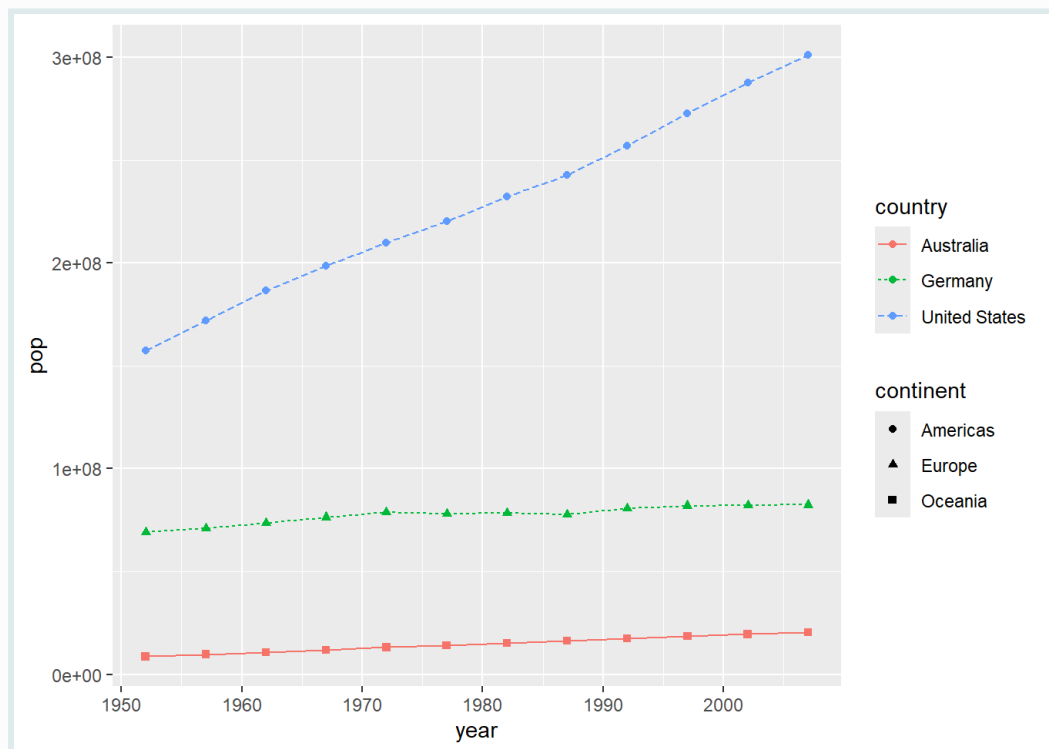
En utilisant le dataframe `gap_mini`, créez un graphique de la croissance de la **population** avec ces mappings esthétiques :



PRACTICE



Ensuite, ajoutez une couche de points au graphique précédent, et ajoutez les mappings esthétiques requis pour produire un graphique qui ressemble à ceci :



PRACTICE



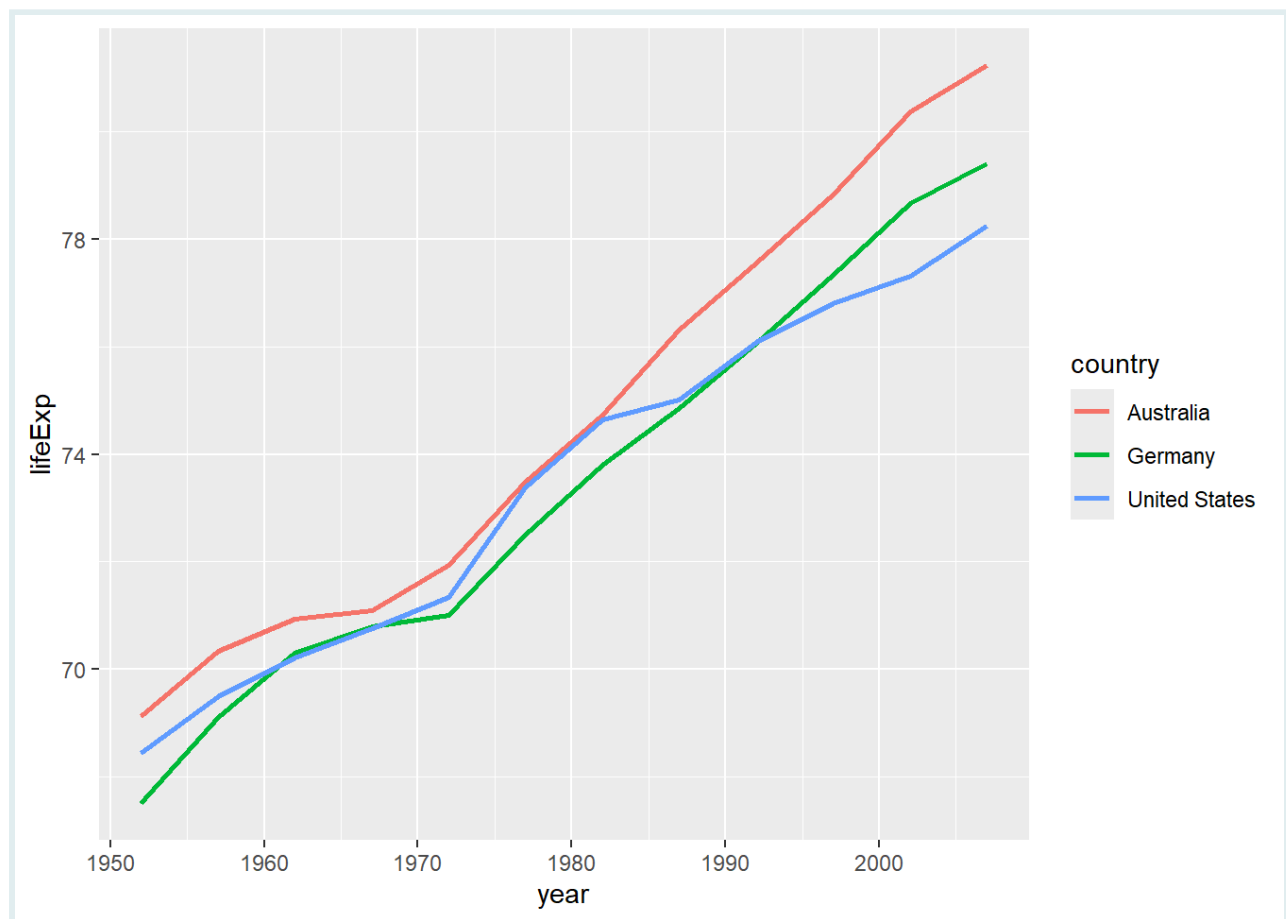
(in RMD)

Ne vous souciez pas des esthétiques fixes, assurez-vous seulement que le mapping des variables est le même.

Modifier les échelles continues x et y

{ggplot2} choisit automatiquement l'échelle à utiliser en fonction du type de variable.

```
# Echelle par défaut pour x, y et color
ggplot(data = gap_mini,
       mapping = aes(x = year,
                     y = lifeExp,
                     color = country)) +
  geom_line(size = 1)
```



Dans certains cas, il faut transformer l'échelle des axes pour une meilleure visualisation. Nous pouvons personnaliser ces échelles avec la famille de fonctions `scale_*`().

```

ggplot(data = <DATAFRAME>,
       mapping = aes(< VARS À MAPPER >)) +
< FONCTION_GEOM > () +
stat = < STAT >, position = < POSITION > ) +
< FONCTION_COORDONNEES > +
< FONCTION_FACET > +
< FONCTION_SCALE > +
< FONCTION_THEME >

```

OBLIGATOIRE

OPTIONNELLE
(VALEUR
PAR DÉFAUT
FOURNIE)

`scale_x_continuous()` et `scale_y_continuous()` sont les fonctions utilisées pour personnaliser les échelle x et y d'un graphique lorsque les données sur ces axes sont continues.

Scales couramment utilisées

A utiliser avec tous paramètres esthétiques:

`scale_*_continuous()` - échelle continue
`scale_*_discrete()` - échelle discrète
`scale_*_identity()` - échelle identité
`scale_*_manual(values = c())` - permet de choisir manuellement les valeurs de l'échelle
`scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks")` - considère les valeurs en tant que date.
`scale_*_datetime()` considère les valeurs de x en tant que datetime. Utilise les mêmes arguments que `scale_x_date()`.

Scales associées à X et Y

A utiliser avec le paramètre esthétique x ou y (exemple ici avec x)

`scale_x_log10()` - échelle logarithmique pour l'axe x
`scale_x_reverse()` - inverse l'axe des x
`scale_x_sqrt()` - échelle « racine carrée » pour l'axe x

Scales de couleur et remplissage (continue)

`o <- a + geom_dotplot(aes(fill = ..x..))`
`o + scale_fill_distiller(palette = «Blues»))`
`o + scale_fill_gradient(low = "red", high = "yellow")`
`o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)`
`o + scale_fill_gradientn(colours = topo.colors(6))`
 voir : `rainbow()`, `heat.colors()`, `terrain.colors()`, `cm.colors()`, `RColorBrewer::brewer.pal()`

Scales de forme et de taille

`p <- e + geom_point(aes(shape = fl, size = cyl))`
`p + scale_shape() + scale_size()`
`p + scale_shape_manual(values = c(3:7))`
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
`p + scale_radius(range = c(1,6))`
`p + scale_size_area(max_size = 6)`

Modifier les graduations des axes

Créons un nouveau subset de pays à partir de `gapminder`. Cette fois, nous allons tracer l'évolution du PIB au fil du temps.

```

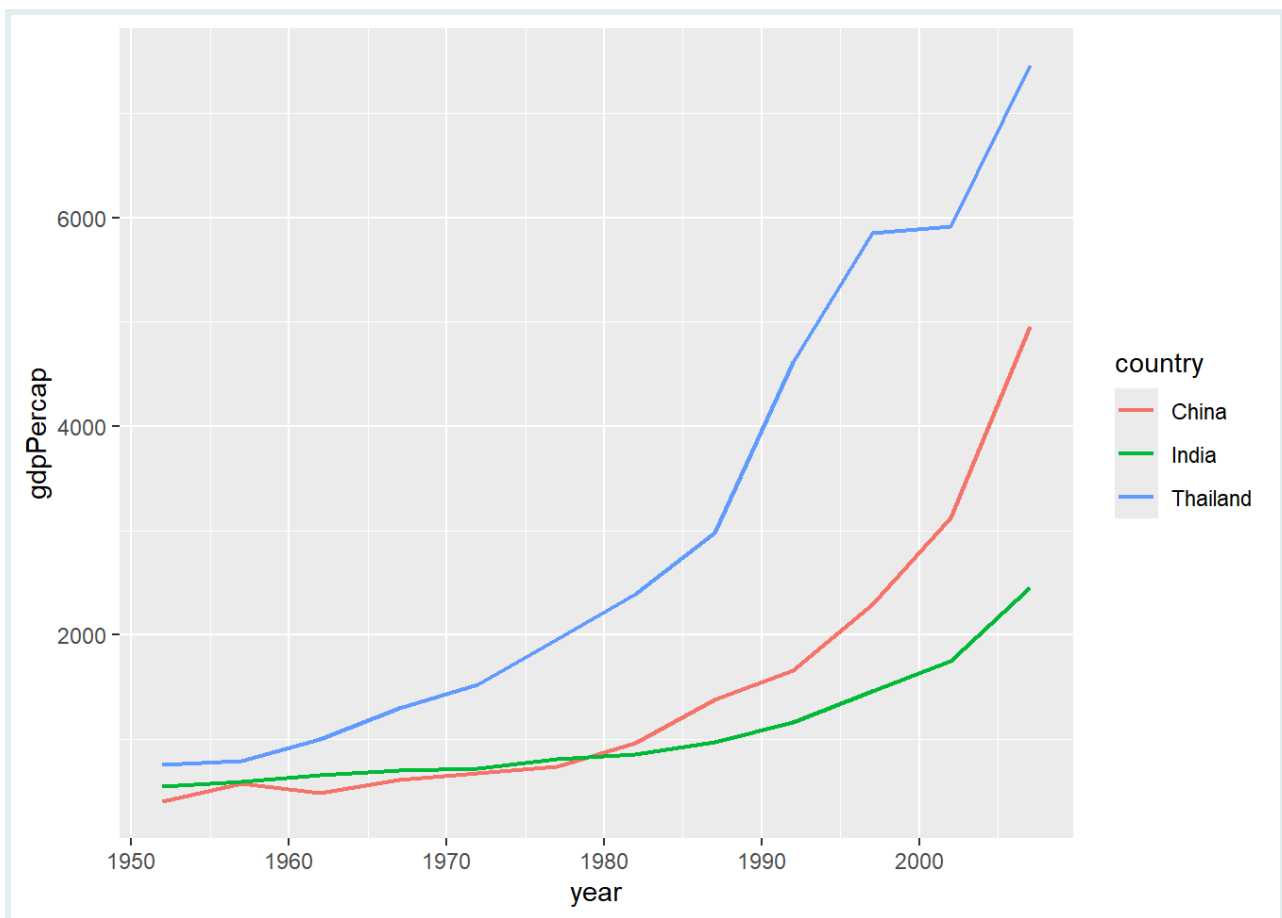
# Subset de données incluant l'Inde, la Chine et la Thaïlande
gap_mini2 <- filter(gapminder,
                    country %in% c("India",

```

```
"China",  
"Thailand"))
```

Ici, nous allons changer le mapping de l'axe des y de `lifeExp` à `gdpPercap` :

```
ggplot(data = gap_mini2,  
       mapping = aes(x = year,  
                     y = gdpPercap,  
                     group = country,  
                     color = country)) +  
geom_line(size = 0.75)
```



Les étiquettes de l'axe des x pour `year` ne correspondent pas aux années dans le dataset.

```
gap_mini2$year %>% unique()
```

```
## [1] 1952 1957 1962 1967 1972 1977 1982 1987  
## [9] 1992 1997 2002 2007
```


Nous pouvons spécifier exactement où étiqueter l'axe en fournissant un vecteur numérique.

```
# Vous pouvez entrer manuellement les graduations (ne faites pas ça)
c(1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002, 2007)
```

```
## [1] 1952 1957 1962 1967 1972 1977 1982 1987
## [9] 1992 1997 2002 2007
```

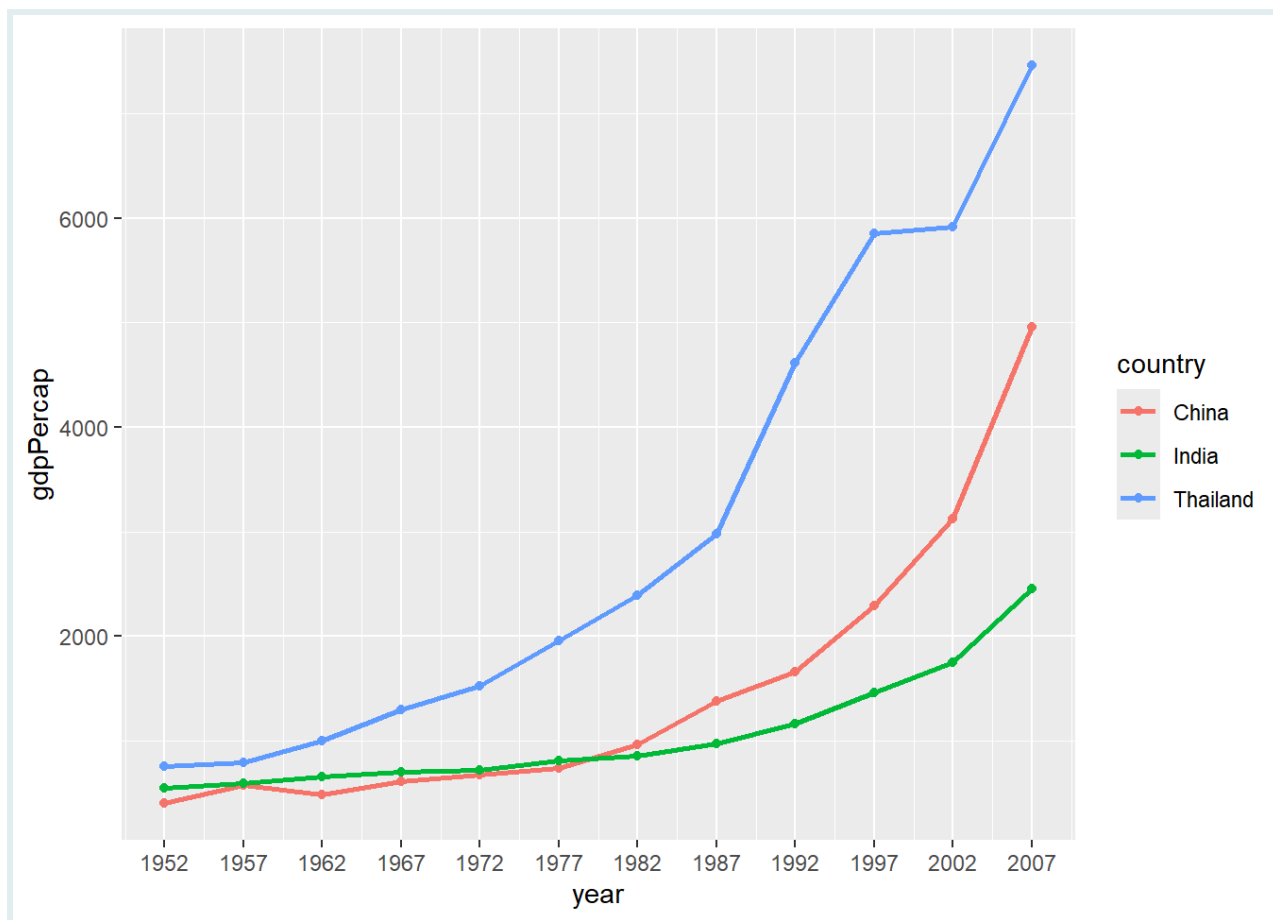
```
# Il est préférable de créer le vecteur avec seq()
seq(from = 1952, to = 2007, by = 5)
```

```
## [1] 1952 1957 1962 1967 1972 1977 1982 1987
## [9] 1992 1997 2002 2007
```

Utilisez `scale_x_continuous` pour faire correspondre les graduations avec le dataset :

```
# Personnalisez les graduations des x avec `scale_x_continuous` (breaks =
  VECTEUR)

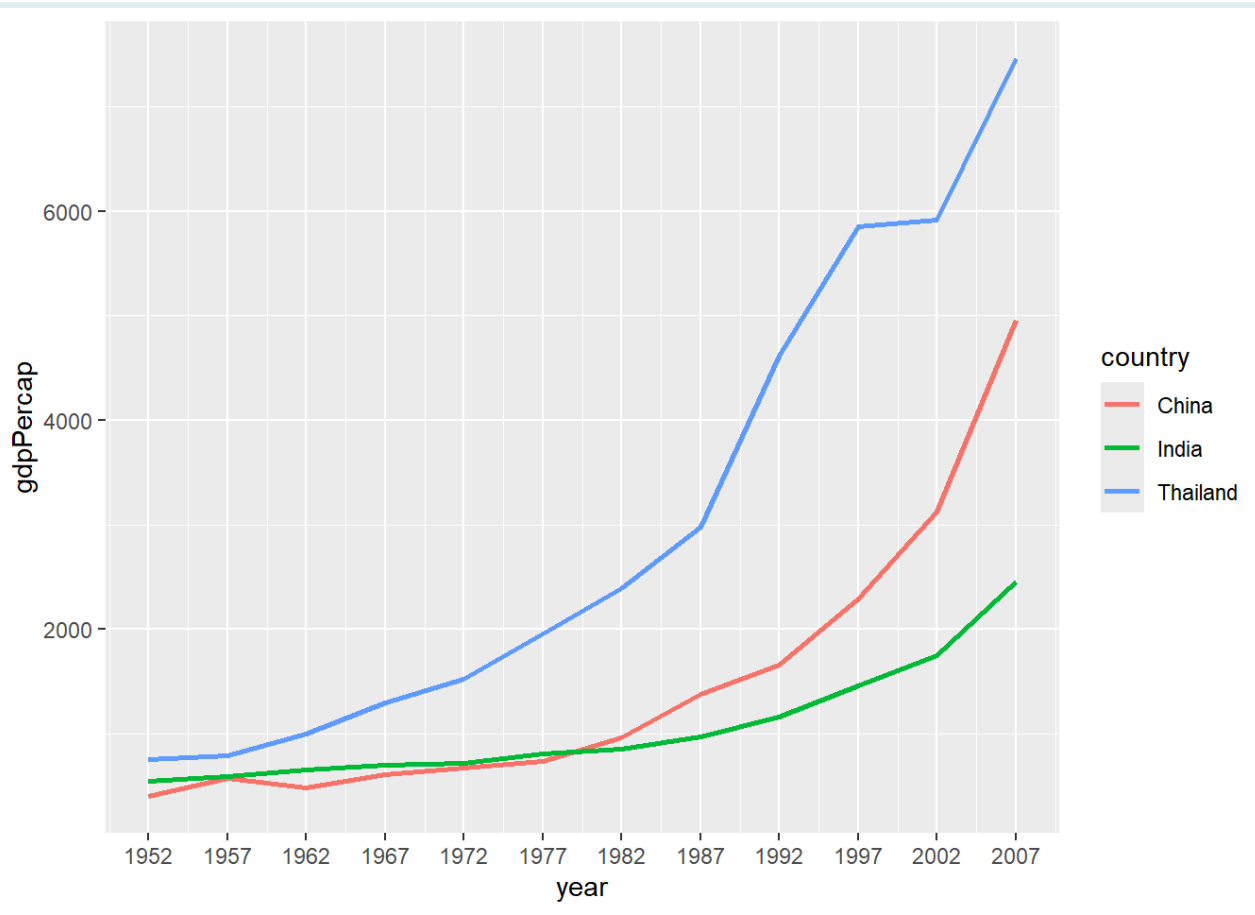
ggplot(data = gap_mini2,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
  geom_line(size = 1) +
  scale_x_continuous(breaks = seq(from = 1952,
                                  to = 2007,
                                  by = 5)) +
  geom_point()
```



Stockez les valeurs des graduations dans un objet R pour les référencer plus facilement.

```
# Stocker le vecteur numérique dans un objet
gap_years <- seq(from = 1952,
                  to = 2007,
                  by = 5)
```

```
# Remplacez le code seq() par l'objet R
ggplot(data = gap_mini2,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
  geom_line(size = 1) +
  scale_x_continuous(breaks = gap_years)
```



PRACTICE

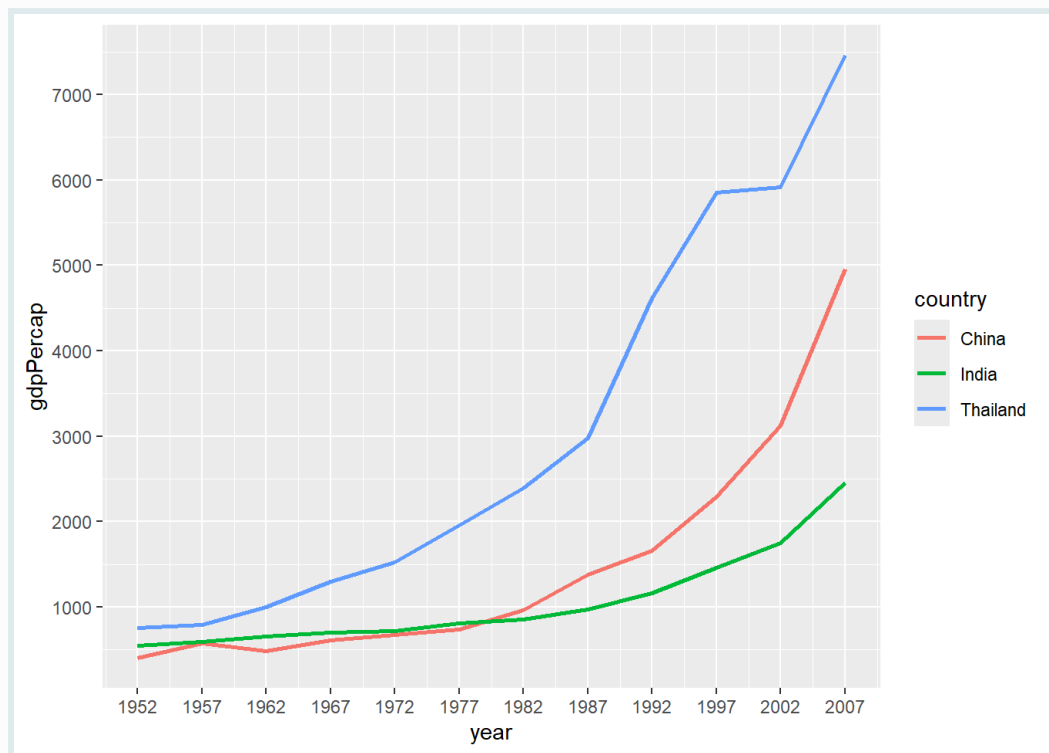


(in RMD)

Nous pouvons personnaliser les graduations de l'axe des **y** continu avec `scale_y_continuous()`.

Copiez le code du dernier exemple et ajoutez `scale_y_continuous()` pour ajouter les graduations de l'axe des **y** suivantes :

PRACTICE



Définir une échelle logarithmique

Dans les deux derniers mini-datasets, nous avons choisi trois pays qui avaient des PIB ou une espérance de vie similaires afin que nous puissions les comparer facilement.

Mais si nous ajoutons un pays qui diffère significativement, l'échelle utilisée par défaut ne convient plus dans ce cas.

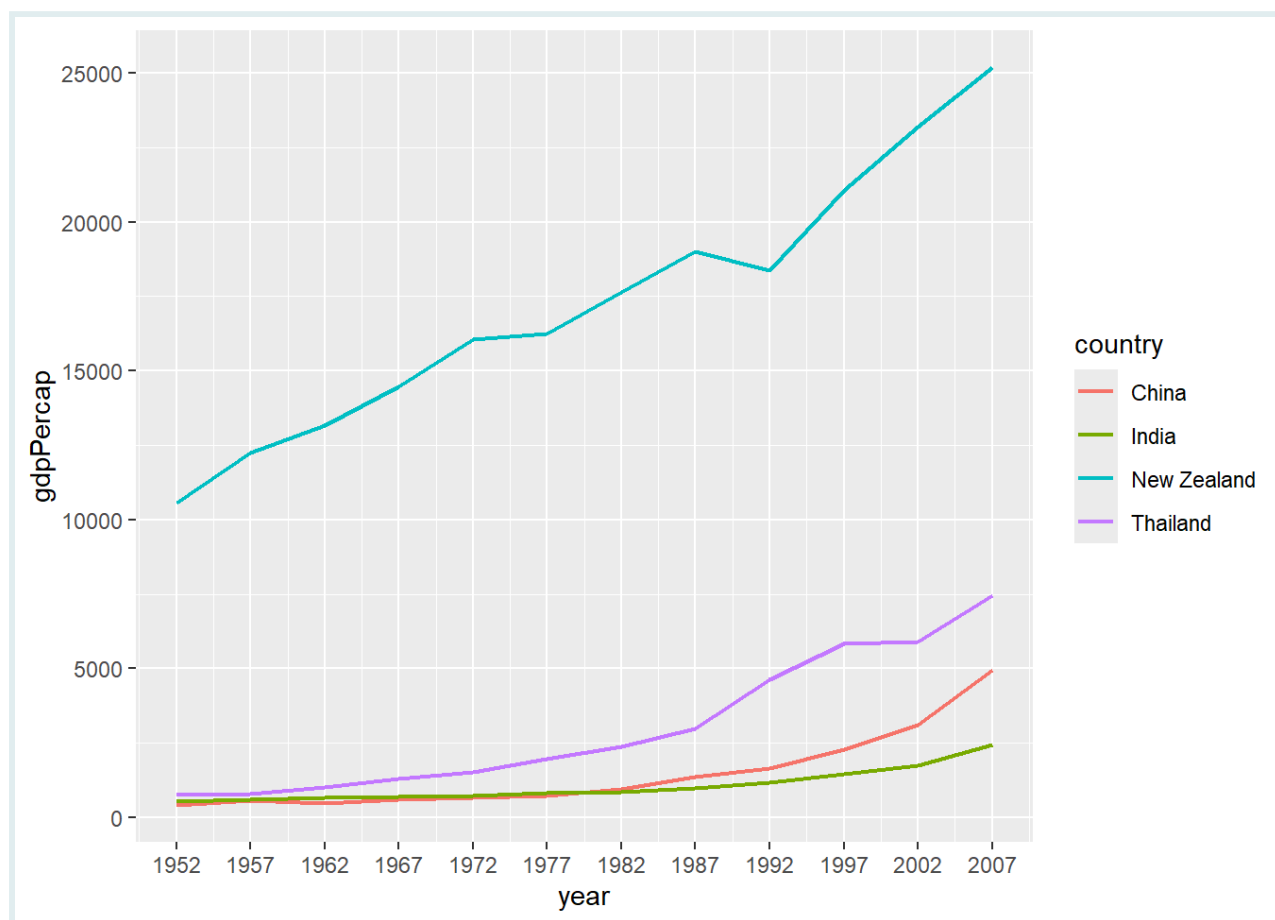
Nous allons voir un exemple où il vaut mieux convertir les axes de l'échelle linéaire par défaut à une échelle logarithmique.

Ajoutons la Nouvelle-Zélande au dataset précédent et créons `gap_mini3` :

```
# Nouveau subset pour inclure l'Inde, la Chine, la Thaïlande et la Nouvelle-  
# Zélande  
gap_mini3 <- filter(gapminder,  
  country %in% c("India",  
                "China",  
                "Thailand",  
                "New Zealand"))  
  
gap_mini3
```

Maintenant, nous allons recréer le graphique du PIB au fil du temps avec le nouveau subset :

```
ggplot(data = gap_mini3,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
geom_line(size = 0.75) +
scale_x_continuous(breaks = gap_years)
```

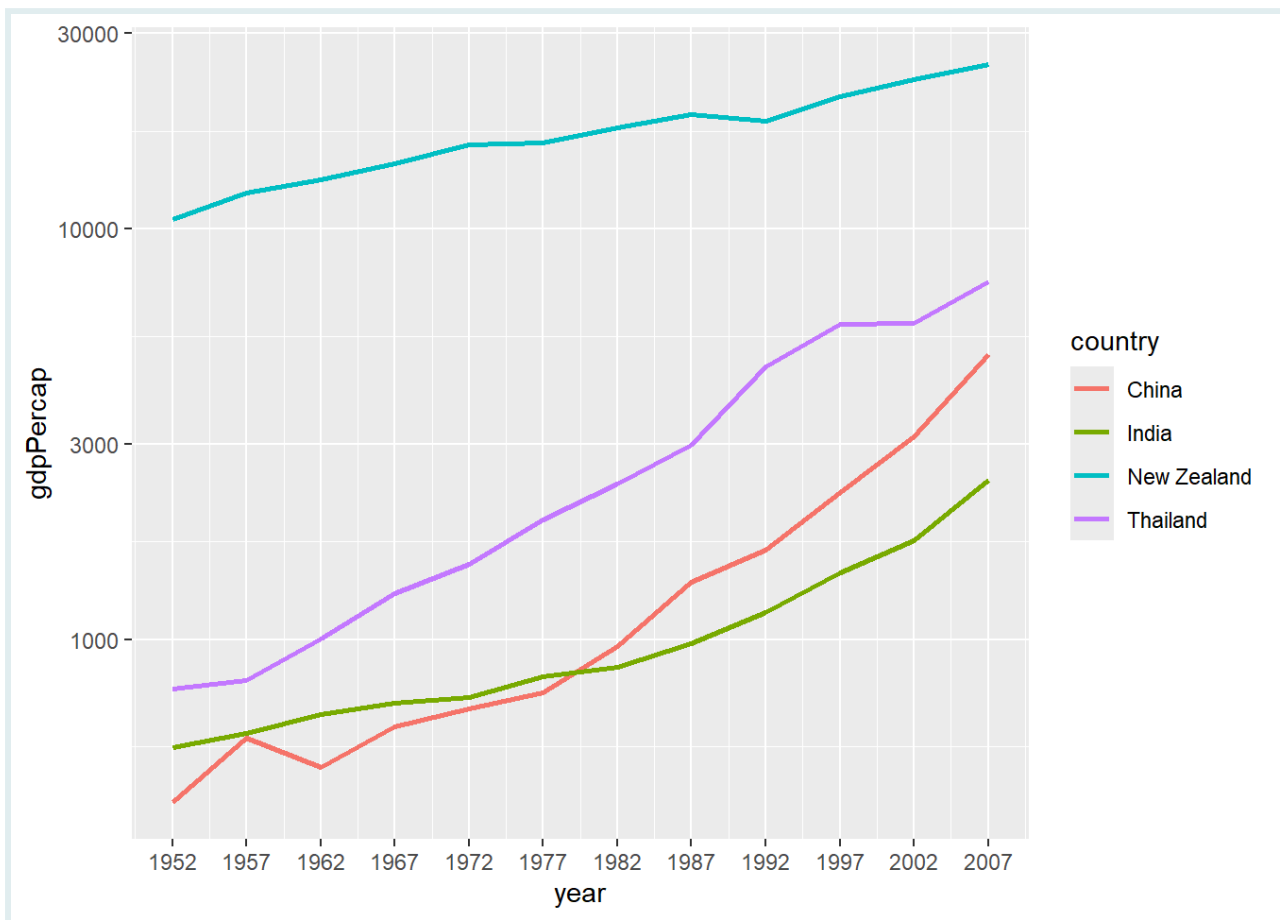


Les courbes pour l'Inde et la Chine montrent une augmentation exponentielle du PIB par habitant. Cependant, les valeurs de l'axe des **y** pour ces deux pays sont beaucoup plus faibles que celle de la Nouvelle-Zélande, donc les lignes sont un peu "tassées". Cela rend les données difficiles à lire. De plus, nous nous retrouvons avec une grande zone vide au milieu.

Nous pouvons résoudre ce problème avec une transformation logarithmique de l'axe des **y** en utilisant `scale_y_log10()`. Nous ajouterons cette fonction en tant que nouvelle couche avec l'opérateur `+`, comme d'habitude :

```
# Ajouter scale_y_log10()
ggplot(data = gap_mini3,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
geom_line(size = 1) +
```

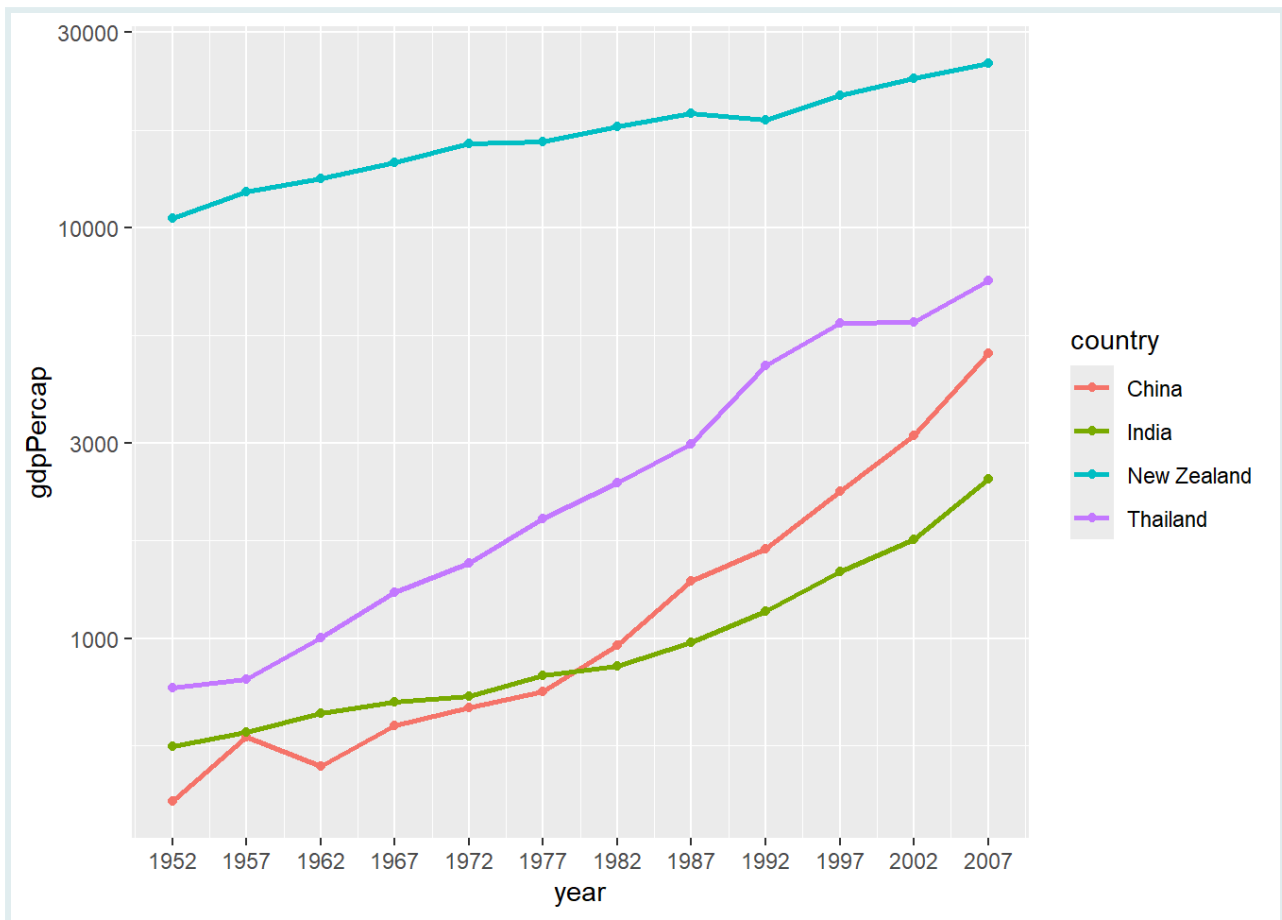
```
scale_x_continuous(breaks = gap_years) + scale_y_log10()
```



Nous avons changé l'échelle de l'axe des **y**, et les étiquettes des graduations d'échelle nous indiquent qu'elle est non linéaire.

Nous pouvons ajouter une couche de points pour rendre cela plus clair :

```
ggplot(data = gap_mini3,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
  geom_line(size = 1) +
  scale_x_continuous(breaks = gap_years) +
  scale_y_log10() +
  geom_point()
```



Tout d'abord, créez un subest à partir de `gapminder` contenant uniquement les données de l'**Uganda** :

Maintenant, utilisez `gap_uganda` pour créer un graphique de série temporelle de la population (`pop`) au fil du temps (`year`). Transformez l'échelle de l'axe des y en une échelle logarithmique, modifiez les graduations en utilisant `gap_years`, changez la couleur de la ligne à `forestgreen` et l'épaisseur à 1mm.

Ensuite, nous pouvons changer le texte des étiquettes des axes pour qu'il soit plus descriptif, et ajouter un titre, un sous-titre et d'autres étiquettes informatives au graphique.

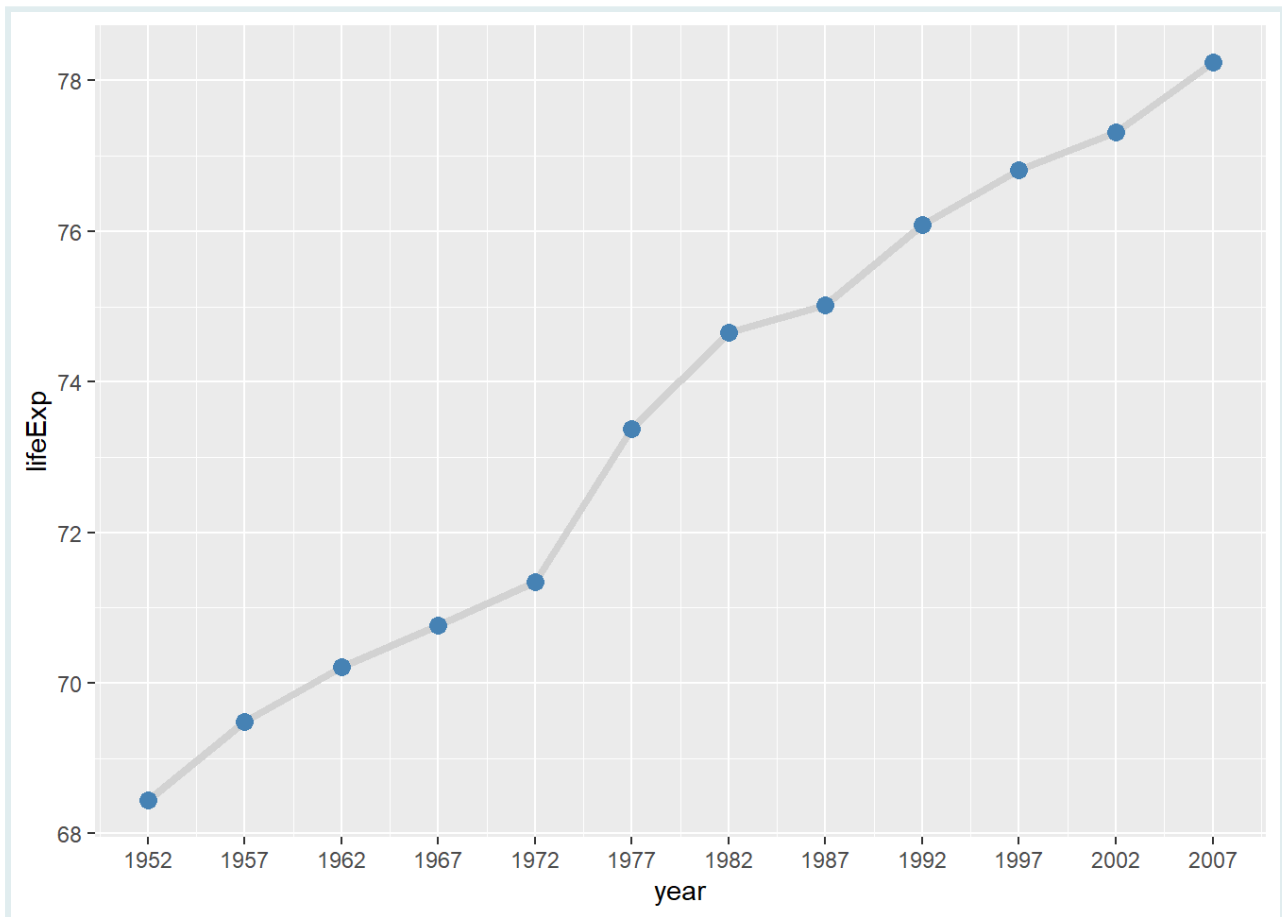
Étiquetage avec `labs()`

Vous pouvez ajouter des étiquettes (labels) à un graphique avec la fonction `labs()`. Les arguments que nous pouvons spécifier avec la fonction `labs()` incluent :

- `title`: Changer ou ajouter un titre
- `subtitle`: Ajouter un sous-titre sous le titre
- `x`: Renommer l'axe des x
- `y`: Renommer l'axe des y
- `caption`: Ajouter une note de bas de page sous le graphique

Commençons par ajouter des étiquettes à ce graphique :

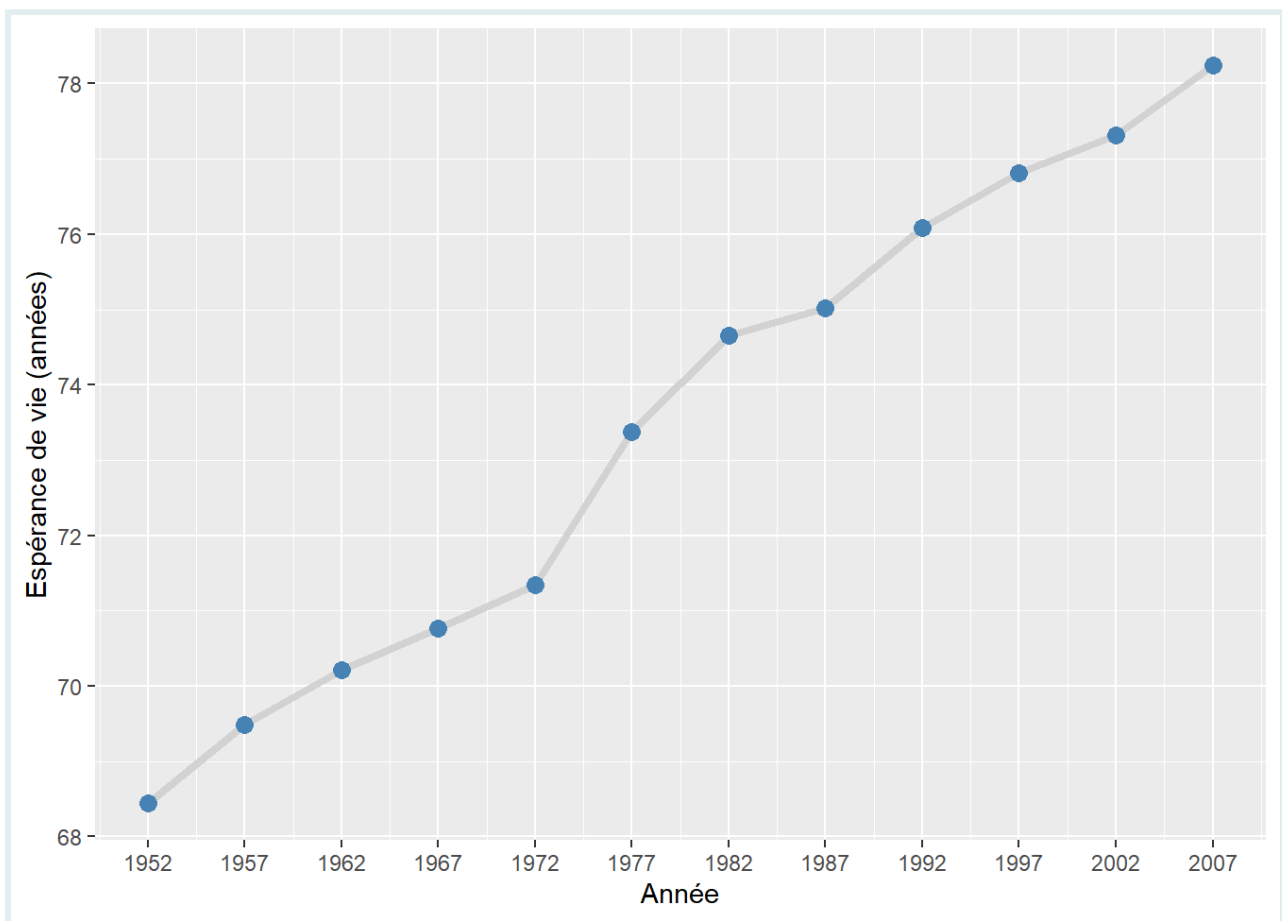
```
# Série temporelle de l'espérance de vie aux États-Unis
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(size = 1.5,
           color = "lightgrey") +
  geom_point(size = 3,
            color = "steelblue") +
  scale_x_continuous(breaks = gap_years)
```



Nous ajoutons `labs()` à notre code en utilisant l'opérateur `+`.

D'abord, nous allons ajouter les arguments `x` et `y` à `labs()`, et changer les titres des axes de la valeur par défaut (nom de la variable) à quelque chose de plus informatif.

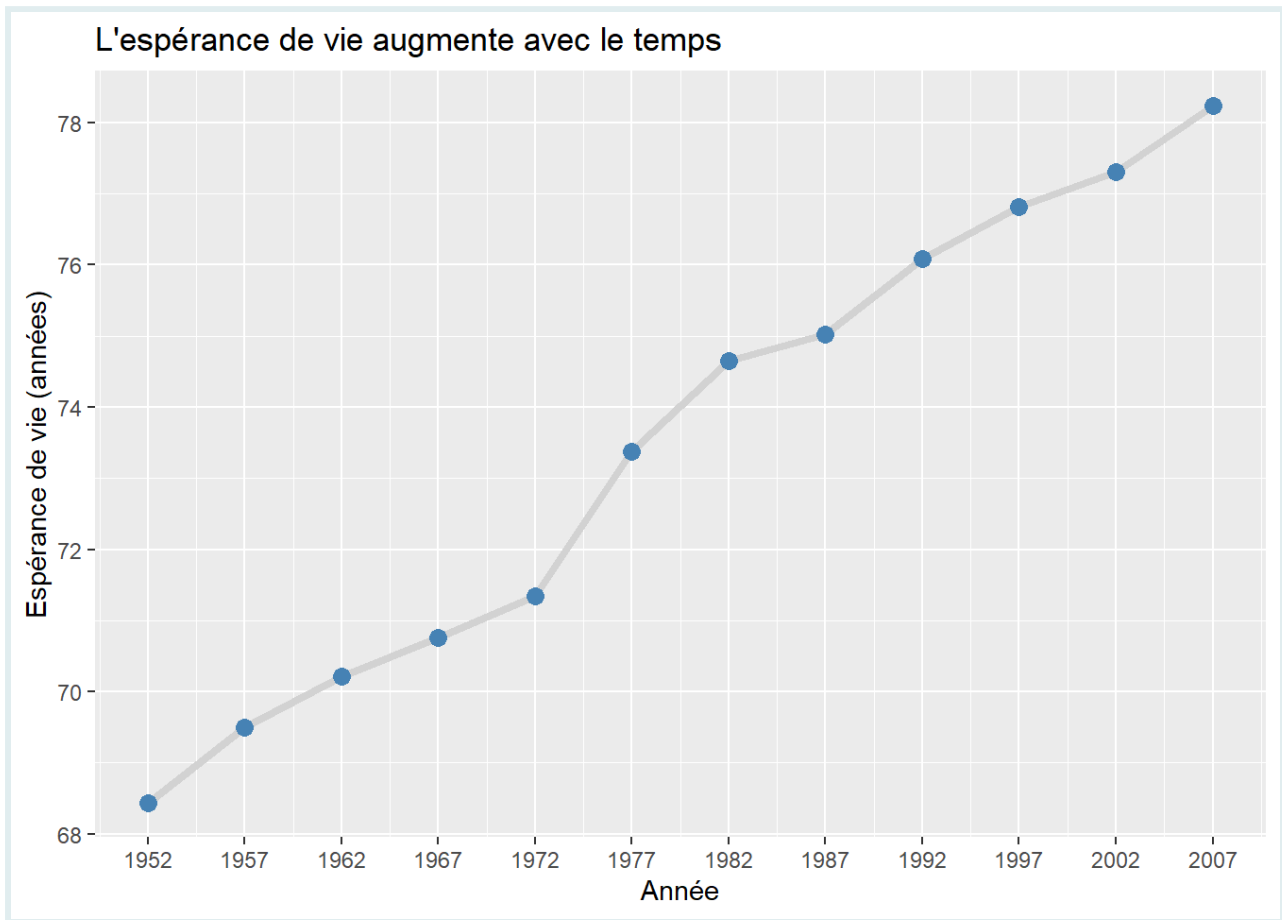

```
# Changer les titres des axes
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(size = 1.5,
           color = "lightgrey") +
  geom_point(size = 3,
            color = "steelblue") +
  scale_x_continuous(breaks = gap_years) +
  labs(x = "Année",
       y = "Espérance de vie (années)")
```



Ensuite, nous allons ajouter un titre au-dessus du graphique avec l'argument `title`.

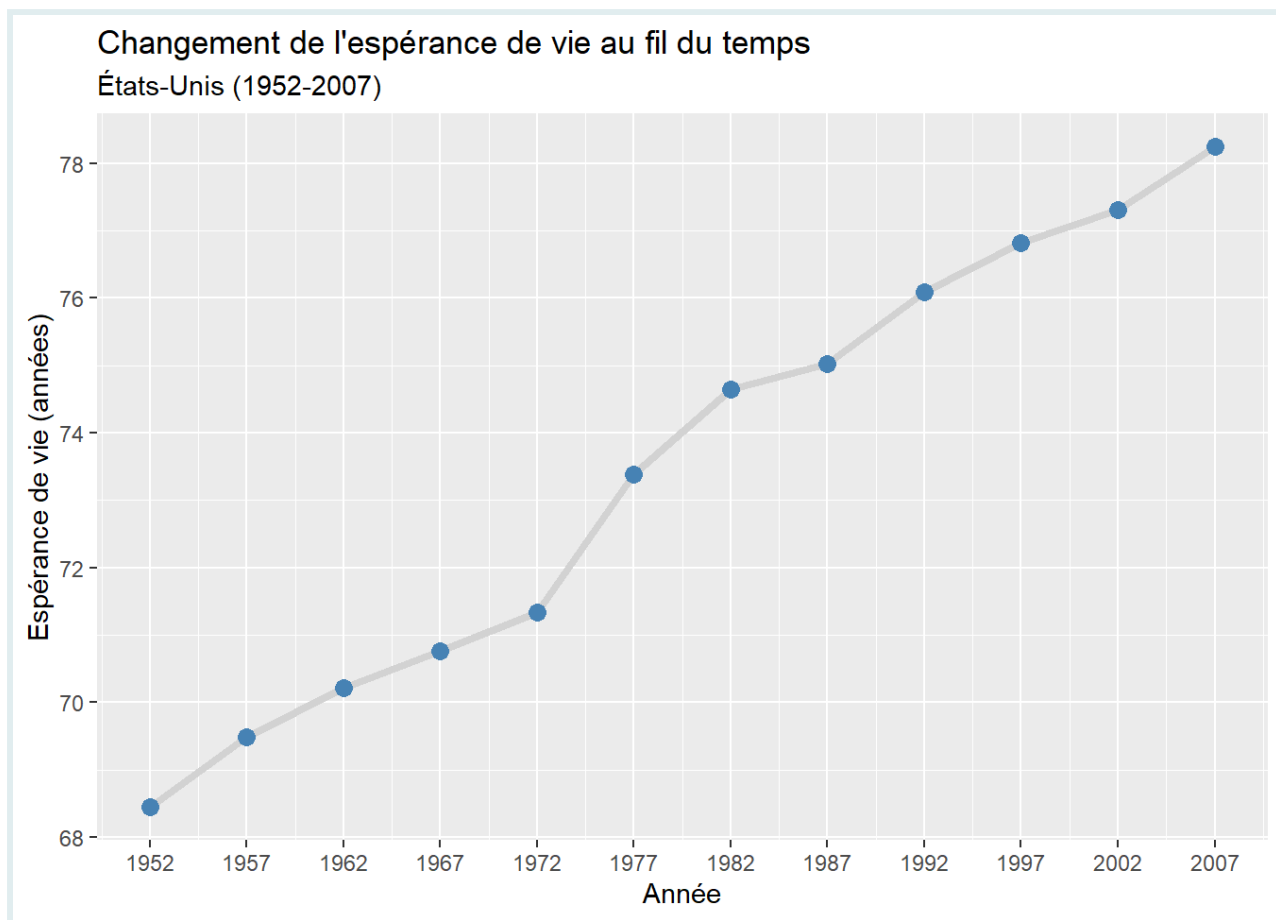
```
# Ajouter un titre principal: "L'espérance de vie augmente avec le temps"
ggplot(data = gap_US,
       mapping = aes(x = year,
                     y = lifeExp)) +
  geom_line(size = 1.5,
           color = "lightgrey") +
  geom_point(size = 3,
            color = "steelblue") +
  scale_x_continuous(breaks = gap_years) +
  labs(x = "Année",
```

```
y = "Espérance de vie (années)",
title = "L'espérance de vie augmente avec le temps")
```



L'argument `subtitle` ajoute un sous-titre sous le titre principal.

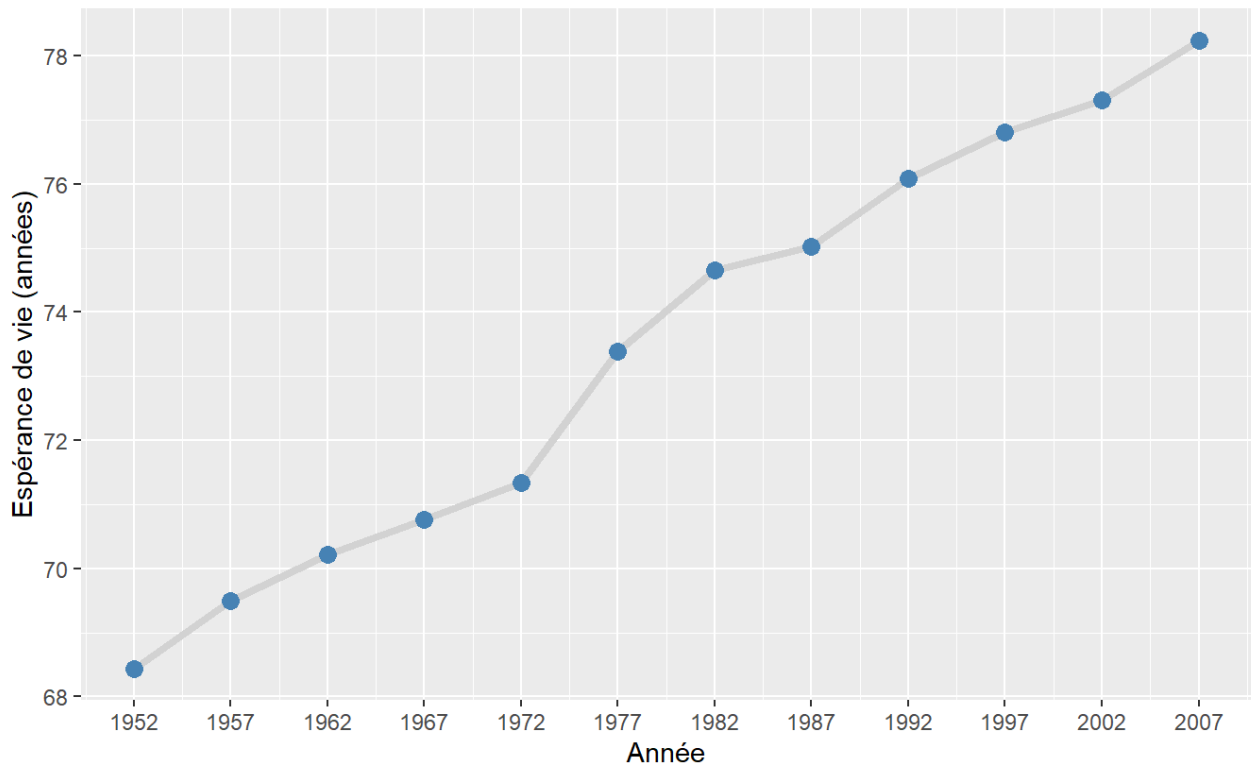
```
# Ajouter un sous-titre avec le lieu et la période
ggplot(data = gap_US,
        mapping = aes(x = year,
                      y = lifeExp)) +
  geom_line(size = 1.5,
            color = "lightgrey") +
  geom_point(size = 3,
             color = "steelblue") +
  scale_x_continuous(breaks = gap_years) +
  labs(x = "Année",
       y = "Espérance de vie (années)",
       title = "Changement de l'espérance de vie au fil du temps",
       subtitle = "États-Unis (1952-2007)")
```



Enfin, nous pouvons fournir l'argument `caption` pour ajouter une note de bas de page sous le graphique.

```
# Ajouter une note de bas de page avec la source des données : "Source :  
  www.gapminder.org/data"  
ggplot(data = gap_US,  
       mapping = aes(x = year,  
                     y = lifeExp)) +  
  geom_line(size = 1.5,  
           color = "lightgrey") +  
  geom_point(size = 3,  
            color = "steelblue") +  
  scale_x_continuous(breaks = gap_years) +  
  labs(x = "Année",  
       y = "Espérance de vie (années)",  
       title = "Changement de l'espérance de vie au fil du temps",  
       subtitle = "États-Unis (1952-2007)",  
       caption = "Source: http://www.gapminder.org/data/")
```

Changement de l'espérance de vie au fil du temps États-Unis (1952-2007)



Source: <http://www.gapminder.org/data/>

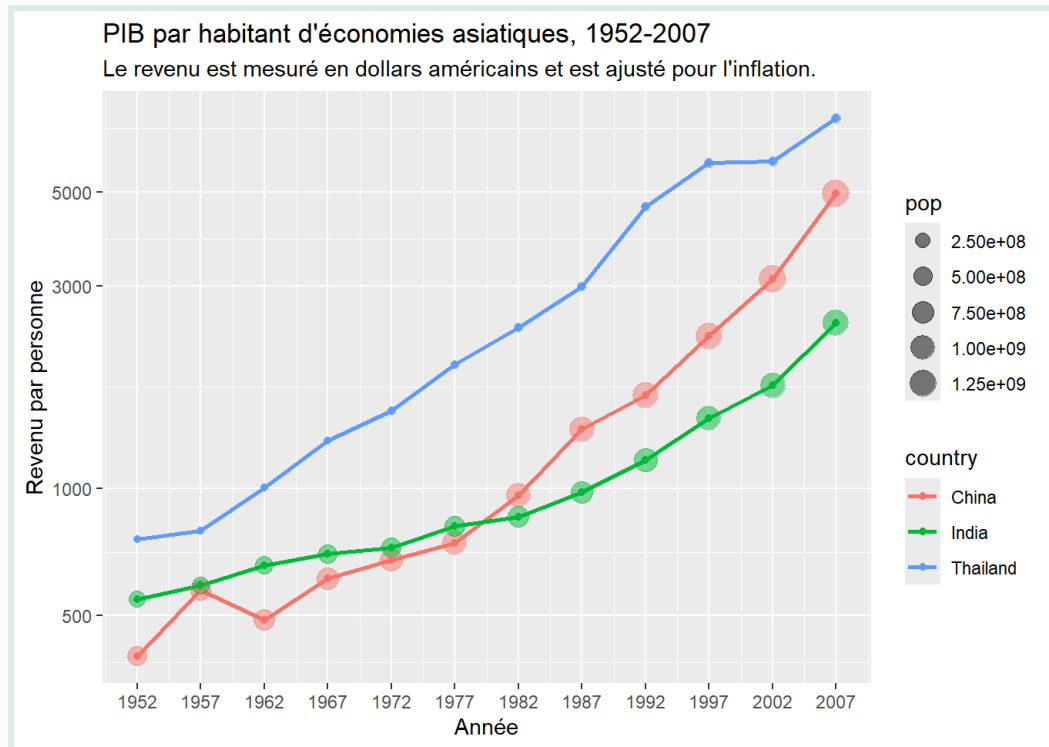
Lorsque vous utilisez un mapping esthétique (par exemple, `color` ou `size`), `{ggplot2}` crée automatiquement une échelle pour cette esthétique en fonction des données fournies et ajoute une légende.

Voici une version mise à jour du graphique `gap_mini3` que nous avons réalisé précédemment. Nous changeons la couleur des points et des lignes en définissant `aes(color = country)` dans `ggplot()`, et la taille des points en définissant `aes(fill = pop)`. Notez que `labs()` est utilisé pour changer le titre, le sous-titre et les étiquettes des axes.



```
ggplot(data = gap_mini2,  
       mapping = aes(x = year,  
                     y = gdpPercap,  
                     color = country)) +  
  geom_line(size = 1) +  
  geom_point(mapping = aes(size = pop),  
            alpha = 0.5) +  
  geom_point() +  
  scale_x_continuous(breaks = gap_years) +  
  scale_y_log10() +
```

```
labs(x = "Année", y = "Revenu par personne",
     title = "PIB par habitant d'économies asiatiques, 1952-2007",
     subtitle = "Le revenu est mesuré en dollars américains et est ajusté pour l'inflation.")
```



Le titre par défaut d'une légende est le nom de la variable à laquelle elle correspond. Ici, la légende de `color` est intitulée `country`, et la légende de `size` est intitulée `pop`.

Nous pouvons également les modifier dans `labs()` en définissant `NOM_AES = "NOUVEAU_TITRE"`.

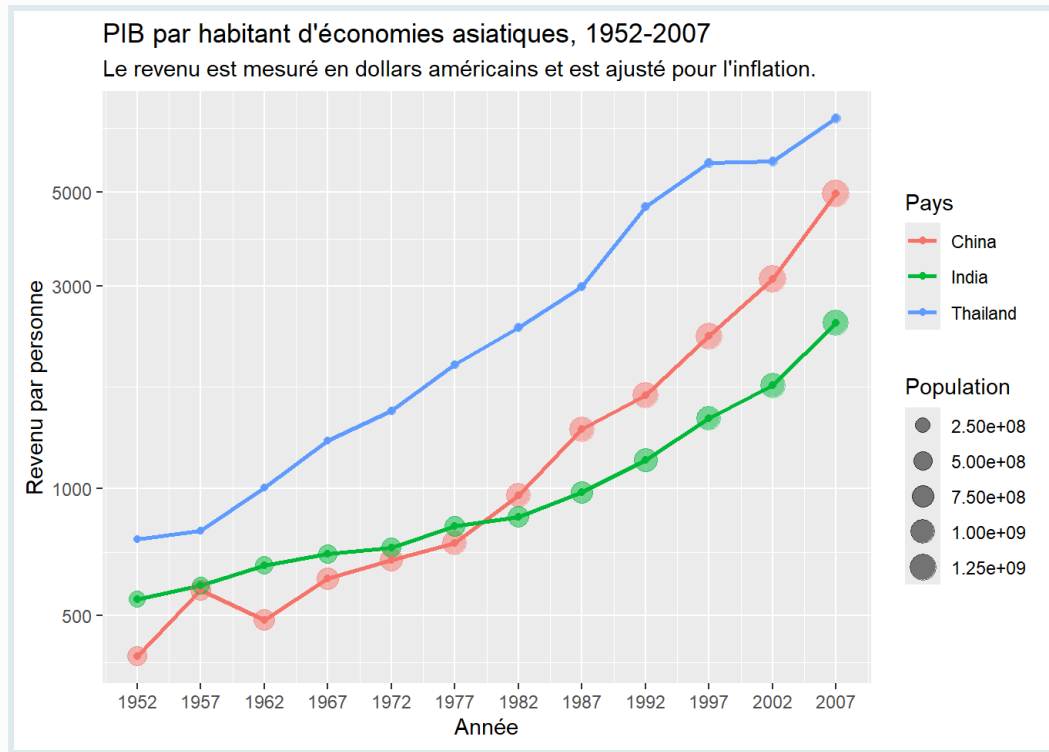
```
ggplot(data = gap_mini2,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
  geom_line(size = 1) +
  geom_point(mapping = aes(size = pop),
            alpha = 0.5) +
  geom_point() +
  scale_x_continuous(breaks = gap_years) +
  scale_y_log10() +
  labs(x = "Année",
       y = "Revenu par personne",
       title = "PIB par habitant d'économies asiatiques, 1952-2007",
```

```

subtitle = "Le revenu est mesuré en dollars américains
           et est ajusté pour l'inflation.",
color = "Pays", size = "Population")

```

CHALLENGE



La même syntaxe peut être utilisée pour modifier les titres des légendes des autres mappings esthétiques. Une erreur courante est d'utiliser le nom de la variable au lieu du nom de l'esthétique dans `labs()`, donc faites attention !

Créez un graphique de série temporelle comparant les tendances du PIB par habitant de 1952 à 2007 pour **trois pays** dans le dataframe `gapminder`.

PRACTICE



(in RMD)

Tout d'abord, créez un sous-ensemble contenant les données de trois pays de votre choix :

Utilisez `my_gap_mini` pour créer un graphique avec les attributs suivants :

- Ajoutez des points au graphique linéaire

- Augmentez l'épaisseur des lignes à 1mm et la taille des points à 2mm
- Rendez les lignes transparentes à 50%
- Changez les intervalles de l'échelle de l'axe des x pour correspondre aux années dans le jeu de données

Enfin, ajoutez les étiquettes suivantes à votre graphique :

- Titre : "Santé & richesse des nations"
- Titres d'axes : "Longévité" et "Année"
- Titre de la légende : "Pays"

PRACTICE



```
# Écrivez le code pour créer votre graphique :
q8 <- "ÉCRIVEZ_VOTRE_CODE_ICI"
```

```
# Vérifiez votre réponse
.CHECK_q8()
```

```
## Incorrect. Votre résultat devrait être un objet ggplot2.
Veuillez réessayer.
## 1 2 3 4 5 6 7 ▷8◁
```

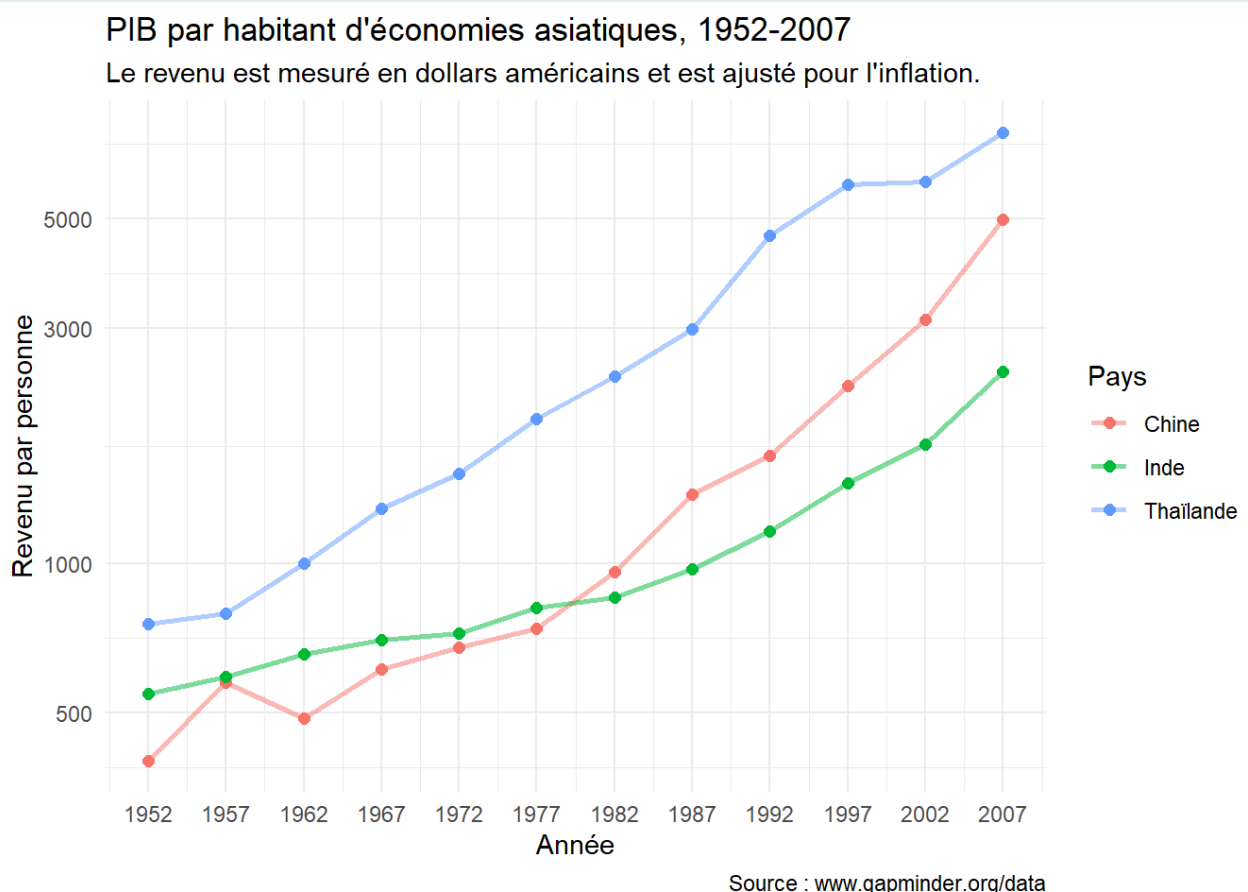
```
.HINT_q8()
```

```
##
## Assurez-vous de ne pas ajouter de modifications
supplémentaires que la question n'a pas demandées, sinon la
fonction CHECK le considérera comme incorrect.
```

Preview : Les thèmes

Dans le prochain cours, nous allons apprendre à utiliser les fonctions `theme`.

```
# Utiliser theme_minimal()
ggplot(data = gap_mini2,
       mapping = aes(x = year,
                     y = gdpPercap,
                     color = country)) +
  geom_line(size = 1, alpha = 0.5) +
  geom_point(size = 2) +
  scale_x_continuous(breaks = gap_years) +
  scale_y_log10() +
  labs(x = "Année",
       y = "Revenu par personne",
       title = "PIB par habitant d'économies asiatiques, 1952-2007",
       subtitle = "Le revenu est mesuré en dollars américains et est ajusté
pour l'inflation.",
       caption = "Source : www.gapminder.org/data",
       color = "Pays") +
  scale_color_discrete(labels = c("China" = "Chine", "India" = "Inde",
                                "Thailand" = "Thaïlande")) +
  theme_minimal()
```



En résumé

Les graphiques linéaires, à l'instar des graphiques de dispersion, sont des outils efficaces pour représenter la relation entre deux variables numériques. Lorsque l'une de ces variables est temporelle, l'usage d'un graphique linéaire est plus adapté. De ce fait, il est conseillé d'opter pour des graphiques linéaires plutôt que des graphiques de dispersion lorsque la variable sur l'axe des abscisses (c'est-à-dire, la variable explicative) possède un ordre intrinsèque, comme c'est le cas pour une variable temporelle telle que `year` dans le dataframe `gapminder`.

Il est possible de transformer les échelles et de modifier les graduations des axes afin de rendre nos graphiques plus lisibles. De plus, l'ajout d'étiquettes permet d'incorporer davantage d'informations.

J'espère que vous avez trouvé ce cours instructif !

Contributeurs

Les membres suivants ont contribué à ce cours :



JOY VAZ

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



IMANE BENSOU DA KORACHI

R Developer and Instructor, the GRAPH Network



ADMIN TEAM

GRAPH Courses Administration Team

The GRAPH Courses team is building epidemiological training courses to enhance disease surveillance and data science for public health across the globe

Références

Le contenu de ce cours a été adapté en partie des sources suivantes :

-
- Ismay, Chester, and Albert Y. Kim. 2022. *A ModernDive into R and the Tidyverse*. <https://moderndive.com/>.
 - Kabacoff, Rob. 2020. *Data Visualization with R*. <https://rkabacoff.github.io/datavis/>.
 - https://www.rebeccabarter.com/blog/2017-11-17-ggplot2_tutorial/

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

