Notes de leçon | Pivoter des données

January 2024

Introduction

Le pivotement ou la mise en forme est une technique de manipulation de données qui consiste à réorienter les lignes et les colonnes d'un ensemble de données. Cela est parfois nécessaire pour faciliter l'analyse des données ou pour rendre les données plus compréhensibles.

Dans cette leçon, nous verrons comment pivoter efficacement les données en utilisant `pivot_longer()` et `pivot_wider()` du package `tidyr`.

Objectifs d'apprentissage

- Vous comprendrez ce que sont le format de données large et le format de données long.
- Vous saurez comment pivoter des données longues en données larges en utilisant `pivot_long()`
- Vous saurez comment pivoter des données larges en données longues en utilisant `pivot_wider()`
- Vous comprendrez pourquoi le format de données long est plus facile pour le tracé et la manipulation dans R.

Packages

```
er les packages
ire(pacman)) install.packages("pacman")
p_load(tidyverse, outbreaks, janitor, rio, here, knitr)
```

Que signifient large et long ?

Les termes large et long sont mieux compris dans le contexte des jeux de données exemples. Examinons-en quelques-uns maintenant.

Imaginez que vous avez trois patients chez qui vous collectez des données de pression artérielle sur trois jours.

Vous pouvez enregistrer les données dans un format large comme celui-ci :

patient	pression_arterielle_jour_1	pression_arterielle_jour_2	pression_arterielle_jour_3
A	110	112	114
B	120	122	124
C	100	104	105

Fig: ensemble de données large pour une série temporelle de patients.

Ou vous pourriez enregistrer les données dans un format long comme suit :

patient	jour	pression_arterielle
A	1	110
A	2	112
A	3	114
B	1	120
B	2	122
B	3	124
C	1	100
C	2	104
C	3	105

Fig: ensemble de données long pour une série temporelle de patients.

Prenez une minute pour étudier les deux ensembles de données pour vous assurer que vous comprenez la relation entre eux.

Dans l'ensemble de données large, chaque unité d'observation (chaque patient) n'occupe qu'une seule ligne. Et chaque mesure (pression artérielle jour 1, pression artérielle jour 2…) se trouve dans une colonne séparée.

Dans l'ensemble de données long, en revanche, chaque unité d'observation (chaque patient) occupe plusieurs lignes, avec une ligne pour chaque mesure.

Voici un autre exemple avec des données fictives, dans lequel les unités d'observation sont des pays :

pays	annee	mesure
x	1960	10
x	1970	13
x	2010	15
y	1960	20
y	1970	23
y	2010	25
z	1960	30
z	1970	33
z	2010	35

Fig: ensemble de données au format long où l'unité d'observation unique est un pays.

pays	annee1960	annee1970	annee2010
x	10	13	15
y	20	23	25
z	30	33	35

Fig: l'ensemble de données au format large équivalent

Les exemples ci-dessus sont tous deux des ensembles de données de séries temporelles, car les mesures sont répétées dans le temps (jour 1, jour 2 et ainsi de suite). Mais les concepts de long et de large sont pertinents pour d'autres types de données également, pas seulement les données de séries temporelles.

Considérez l'exemple ci-dessous, qui montre le nombre de patients dans différentes unités de trois hôpitaux :

Hopital	Unite maternite	Unite soins intensifs
Hopital A	4	2
Hopital B	5	2
Hopital C	6	3

Fig: ensemble de données au format large, où chaque hôpital est une unité d'observation

Hopital	Unite	Nbr. de patients
Hopital A	Maternite	4
Hopital A	Soins intensifs	2
Hopital B	Maternite	5
Hopital B	Soins intensifs	2
Hopital C	Maternite	6
Hopital C	Soins intensifs	3

Fig: l'ensemble de données au format long équivalent

Dans l'ensemble de données au format large, encore une fois, chaque unité d'observation (chaque hôpital) n'occupe qu'uneseule ligne, avec les mesures répétées pour cette unité (nombre de patients dans différentes salles) réparties sur deux colonnes.

Dans l'ensemble de données au format long, chaque unité d'observation est répartie sur plusieurs lignes.

Les "unités d'observation", parfois appelées "unités statistiques" d'un ensemble de données, sont les entités ou les éléments principaux décrits par les colonnes de cet ensemble de données.

Dans le premier exemple, les unités d'observation/statistiques étaient les patients ; dans le deuxième exemple, les pays, et dans le troisième exemple, les hôpitaux.

Pratique 1

Considérez l'ensemble de données fictif créé ci-dessous :

```
tures <-
frame(
 = c("Suède", "Danemark", "Norvège"),
moy.1994 = 1:3,
moy.1995 = 3:5,
moy.1996 = 5:7)
tures
```


```
##       pays tempmoy.1994 tempmoy.1995 tempmoy.1996
## 1    Suède            1            3            5
## 2 Danemark            2            4            6
## 3  Norvège            3            5            7
```

Ces données sont-elles dans un format large ou long ?

```
 la chaîne "large" ou la chaîne "long"
'
```

~~Quand devez~~-vous utiliser des données en format large ou long ?

La vérité est que cela dépend vraiment de ce que vous voulez faire ! Le format large est parfait pour *afficher les données* car il est facile de comparer visuellement les valeurs de cette manière. Le format long est idéal pour certaines tâches d'analyse de données, comme le regroupement et la conception des graphiques.

Il sera donc essentiel pour vous de savoir comment passer facilement de l'un à l'autre format. Passer du format large au format long, ou vice versa, est appelé **pivotement**.

Pivoter de large à long

Pour pratiquer le pivotement d'un format large à un format long, nous allons considérer les données de Gapminder sur le **nombre de décès d'enfants en bas âge** dans certains pays au fil des ans.

SIDE NOTE

collections.

Ci-dessous, nous lisons et visualisons ces données sur les décès d'enfants en bas âge :

```
ıfants_large <- read_csv(here("data/fr_gapminder_infant_deaths.csv"))
ıfants_large
```

```
## # A tibble: 5 × 7
##   pays                  x2010 x2011 x2012 x2013 x2014 x2015
##   <chr>                 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan           74600 72000 69500 67100 64800 62700
## 2 Angola                79100 76400 73700 71200 69000 67200
## 3 Albania                 420   384   354   331   313   301
## 4 United Arab Emirates    683   687   686   681   672   658
## 5 Argentina              9550  9230  8860  8480  8100  7720
```

Nous observons que chaque unité d'observation (chaque pays) occupe une seule ligne, les mesures répétées étant réparties sur plusieurs colonnes. Par conséquent, cet ensemble de données est dans un format large.

Pour convertir en format long, nous pouvons utiliser une fonction pratique `pivot_longer`. Dans `pivot_longer`, nous définissons, à l'aide de l'argument `cols`, les colonnes que nous voulons pivoter :

```
ıfants_large %>%
_longer(cols = x2010:x2015)
```

```
## # A tibble: 5 × 3
##   pays        name  value
##   <chr>       <chr> <dbl>
## 1 Afghanistan x2010 74600
## 2 Afghanistan x2011 72000
## 3 Afghanistan x2012 69500
## 4 Afghanistan x2013 67100
## 5 Afghanistan x2014 64800
```

Très facile !

Nous pouvons observer que l'ensemble de données au format long résultant a chaque pays occupant 5 lignes (une par année entre 2010 et 2015). Les années sont indiquées dans la variable `names`, et toutes les valeurs de décompte de décès occupent une seule variable, `values`.

Une façon utile de penser à cette transformation est que les valeurs des décès d'enfants étaient auparavant en format matriciel (2 dimensions ; 2D), mais qu'elles sont maintenant en format vectoriel (1 dimension ; 1D).

Cet ensemble de données long sera beaucoup plus pratique pour de nombreuses procédures d'analyse de données.

En tant que bon analyste de données, vous pouvez trouver que les noms par défaut des variables, `names` et `values`, ne sont pas satisfaisants ; ils ne décrivent pas adéquatement ce que contiennent les variables. Pas de souci ; vous pouvez donner des noms de colonnes personnalisés, en utilisant les arguments `names_to` et `values_to` :

```
ıfants_large %>%
_longer(cols= x2010:x2015,
        names_to = "année",
        values_to = "nombre_deces")
```

```
## # A tibble: 5 × 3
##   pays        année nombre_deces
##   <chr>       <chr>        <dbl>
## 1 Afghanistan x2010        74600
## 2 Afghanistan x2011        72000
## 3 Afghanistan x2012        69500
## 4 Afghanistan x2013        67100
## 5 Afghanistan x2014        64800
```

Remarquez que le format long est plus informatif que le format large original. Pourquoi ? À cause du nom de colonne informatif "nombre_deces". Dans le format large, à moins que le CSV ne soit nommé quelque chose comme `nombre_deces_enfants`, ou que quelqu'un vous dise "ce sont les comptes de décès d'enfants par pays et par année", vous n'avez aucune idée de ce que représentent les chiffres dans les cellules.

Vous voudrez peut-être aussi supprimer le `x` devant chaque année. Cela peut être réalisé avec la fonction pratique `parse_number()` du package {readr} (partie de tidyverse), qui extrait les chiffres des chaînes :

```
ıfants_large %>%
_longer(cols = x2010:x2015,
        names_to = "année",
        values_to = "nombre_deces") %>%
e(année = parse_number(année))
```

```
## # A tibble: 5 × 3
##   pays        année nombre_deces
##   <chr>       <dbl>        <dbl>
## 1 Afghanistan  2010        74600
## 2 Afghanistan  2011        72000
## 3 Afghanistan  2012        69500
## 4 Afghanistan  2013        67100
## 5 Afghanistan  2014        64800
```

Super ! Nous avons maintenant un ensemble de données au format long et propres.

Pour une utilisation ultérieure, enregistrons maintenant ces données :

```
nfants_long <-
_enfants_large %>%
_longer(cols = x2010:x2015,
        names_to = "annee",
        values_to = "nombre_deces")
```

Pratique 2

Pour cette question de pratique, vous utiliserez le jeu de données `euro_births_wide` provenant d'Eurostat. Il montre le nombre annuel de naissances dans 50 pays européens :

```
es_euro_large <-
sv(here("data/fr_euro_births_wide.csv"))
ssances_euro_large)
```


```
## # A tibble: 5 × 8
##   pays      x2015  x2016  x2017  x2018  x2019  x2020  x2021
##   <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Belgium  122274 121896 119690 118319 117695 114350 118349
## 2 Bulgaria  65950  64984  63955  62197  61538  59086  58678
## 3 Czechia  110764 112663 114405 114036 112231 110200 111793
## 4 Denmark   58205  61614  61397  61476  61167  60937  63473
## 5 Germany  737575 792141 784901 787523 778090 773144 795492
```

Les données sont dans un format large. Convertissez-le en un jeu de données au format long qui a les noms de colonnes suivants : "pays", "annee" et "nombre_de_naissances"

```
ances_euro_large %>% # complétez le code avec votre réponse
```

Pivoter de long à large

Maintenant que vous savez comment pivoter de large à long avec `pivot_longer()`. Comment faire l'inverse, de long à large ? Pour cela, vous pouvez utiliser la fonction bien nommée `pivot_wider()`.

Mais avant de considérer comment utiliser cette fonction pour manipuler des données longues, considérons d'abord *où* vous êtes susceptible de rencontrer des données longues.

Alors que les données larges proviennent généralement de sources externes (comme nous l'avons vu ci-dessus), les données longues, en revanche, sont susceptibles d'être créées par *vous* lors de la manipulation des données, notamment lors des manipulations `group_by()`-`summarize()`.

Voyons un exemple maintenant.

Nous utiliserons un jeu de données de dossiers de patients provenant d'une épidémie d'Ebola en Sierra Leone en 2014. Ci-dessous, nous extrayons ces données du package {outbreaks} et effectuons quelques manipulations de simplification.

```
eaks::ebola_sierraleone_2014 %>%
bble() %>%
e(annee = lubridate::year(date_of_onset)) %>% # extraire l'année de la date
t(id_patient = id, district, annee_de_debut = annee) # sélectionner et renommer
```

```
## # A tibble: 5 × 3
##   id_patient district annee_de_debut
##        <int> <fct>             <dbl>
## 1          1 Kailahun           2014
## 2          2 Kailahun           2014
## 3          3 Kailahun           2014
## 4          4 Kailahun           2014
## 5          5 Kailahun           2014
```

Chaque ligne correspond à un patient, et nous avons le numéro d'identifiant de chaque patient, leur district et l'année où ils ont contracté Ebola.

Maintenant, considérez la résumé groupé suivant du jeu de données `ebola`, qui compte le nombre de patients enregistrés dans chaque district chaque année :

```
district_par_annee <-
 %>%
by(district) %>%
(annee_de_debut) %>%
p()

district_par_annee
```

```
## # A tibble: 5 × 3
##   district annee_de_debut     n
##   <fct>             <dbl> <int>
## 1 Bo                 2014   397
## 2 Bo                 2015   209
## 3 Bombali            2014  1070
## 4 Bombali            2015   120
## 5 Bonthe             2014     7
```

La sortie de cette opération groupée est un jeu de données typiquement “long” ! Chaque unité d’observation (chaque district) occupe plusieurs lignes (deux lignes par district, pour être exact), avec une ligne pour chaque mesure (chaque année).

Ainsi, comme vous le voyez maintenant, les données longues peuvent souvent arriver comme une sortie de résumés groupés, entre autres manipulations de données.

Maintenant, voyons comment convertir de telles données longues en un format large avec `pivot_wider()`.

Le code est assez simple :

```
district_par_annee %>%
wider(values_from = n,
      names_from = annee_de_debut)
```

```
## # A tibble: 5 × 3
##   district `2014` `2015`
##   <fct>     <int>  <int>
## 1 Bo          397    209
## 2 Bombali    1070    120
## 3 Bonthe        7     77
## 4 Kailahun    535     35
## 5 Kambia      127    294
```

Comme vous pouvez le voir, `pivot_wider()` a deux arguments importants : `values_from` et `names_from`. L’argument `values_from` définit quelles valeurs deviendront le cœur du format de données large (en d’autres termes : quel vecteur 1D deviendra une matrice 2D). Dans notre cas, ces valeurs étaient dans la variable `n`. Et `names_from` identifie quelle variable utiliser pour définir les noms de colonnes dans le format large. Dans notre cas, c’était la variable `annee_de_début`.

Vous voudrez peut-être aussi faire des *années* votre unité d'observation/statistique principale, avec chaque année occupant une ligne. Cela peut être réalisé de manière similaire à l'exemple ci-dessus, mais la variable `district` sera fournie en argument à `names_from`, au lieu de `annee_de_debut`.

```
district_par_annee %>%
wider(values_from = n,
      names_from = district)
```

SIDE NOTE

```
## # A tibble: 2 × 15
##   annee_de_debut    Bo Bombali Bonthe Kailahun Kambia
Kenema
##            <dbl> <int>   <int>  <int>    <int>  <int>
<int>
## 1           2014   397    1070      7      535    127
641
## 2           2015   209     120     77       35    294
139
## # ℹ 8 more variables: Koinadugu <int>, Kono <int>,
## #   Moyamba <int>, `Port Loko` <int>, Pujehun <int>, …
```

Ici, les unités d'observation uniques (nos lignes) sont maintenant les années (2014, 2015).

Pratique 3

PRACTICE

(in RMD)

L'ensemble de données `population` du package `tidyr` montre les populations de 219 pays au fil du temps.

Pivoter ces données en un format large. Votre réponse devrait comprendre 20 colonnes et 219 lignes.

```
:population %>%
```

Pourquoi les données au format long sont-elles meilleures pour l'analyse?

Ci-dessus, nous avons mentionné que les données au format long sont les meilleures pour la majorité des tâches d'analyse de données. Maintenant, nous pouvons justifier pourquoi. Dans les sections ci-dessous, nous passerons en revue quelques opérations courantes que vous devrez effectuer avec des données au format long. Dans chaque cas, vous observerez que les manipulations similaires sur des données larges seraient assez délicates.

Filtrer les données groupées

Tout d'abord, parlons du filtrage des données groupées, qui est très facile à faire sur des données au format long, mais difficile sur des données au format large.

Voici un exemple avec le jeu de données sur la mortalité infantile. Imaginez que nous voulons répondre à la question suivante : **Pour chaque pays, quelle année a connu le plus grand nombre de décès d'enfants ?**

Voici comment nous le ferions avec le format long des données :

```
nfants_long %>%
_by(pays) %>%
(nombre_deces == max(nombre_deces))
```

```
## # A tibble: 5 × 3
## # Groups:   pays [5]
##   pays                  annee nombre_deces
##   <chr>                 <chr>        <dbl>
## 1 Afghanistan           x2010        74600
## 2 Angola                x2010        79100
## 3 Albania               x2010          420
## 4 United Arab Emirates  x2011          687
## 5 Argentina             x2010         9550
```

Facile, n'est-ce pas ? Nous pouvons facilement voir, par exemple, que l'Afghanistan a eu son plus haut taux de mortalité infantile en 2010, et les Émirats arabes unis en 2011.

Si vous vouliez faire la même chose avec des données au format large, ce serait beaucoup plus difficile. Vous pourriez essayer une approche comme celle-ci avec `rowwise()` :

```
nfants_large %>%
se() %>%
(max_decompte = max(x2010, x2011, x2012, x2013, x2014, x2015))
```

```
## # A tibble: 5 × 8
## # Rowwise:
##   pays                  x2010 x2011 x2012 x2013 x2014 x2015
##   <chr>                 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan           74600 72000 69500 67100 64800 62700
## 2 Angola                79100 76400 73700 71200 69000 67200
## 3 Albania                 420   384   354   331   313   301
## 4 United Arab Emirates    683   687   686   681   672   658
## 5 Argentina              9550  9230  8860  8480  8100  7720
## # ℹ 1 more variable: max_decompte <dbl>
```

Cela fonctionne presque - nous avons, pour chaque pays, le nombre maximum de décès d'enfants rapportés - mais nous ne savons toujours pas à quelle année est attachée cette valeur dans `max_decompte`. Nous devrions prendre cette valeur et l'indexer à sa colonne d'année respective d'une manière ou d'une autre… quelle corvée ! Il y a des solutions pour trouver cela mais toutes sont très pénibles. Pourquoi rendre votre vie compliquée quand vous pouvez simplement pivoter vers le format long et utiliser la beauté de `group_by()` et `filter()` ?

SIDE NOTE

Ici, nous avons utilisé une fonction spéciale de {dplyr} : `rowwise()`. `rowwise()` permet d'appliquer d'autres opérations *par ligne*. Il est équivalent à la création d'un groupe pour chaque ligne (`group_by(row_number())`).

Sans `rowwise()`, vous obtiendriez ceci :

```
ıfants_large %>%
ɐ(max_decompte = max(x2010, x2011, x2012, x2013, x2014, x2015))
```

```
## # A tibble: 5 × 8
##   pays                  x2010 x2011 x2012 x2013 x2014 x2015
##   <chr>                 <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan           74600 72000 69500 67100 64800 62700
## 2 Angola                79100 76400 73700 71200 69000 67200
## 3 Albania                 420   384   354   331   313   301
## 4 United Arab Emirates    683   687   686   681   672   658
## 5 Argentina              9550  9230  8860  8480  8100  7720
## # ℹ 1 more variable: max_decompte <dbl>
```

… le compte maximum sur TOUTES les lignes de l'ensemble de données.

PRACTICE

Pratique 4

Pour cette question de pratique, vous effectuerez un filtre groupé sur le jeu de données `population` en format long du package `tidyr`. Utilisez `group_by()` et `filter()` pour obtenir un ensemble de données qui montre la population maximale enregistrée pour chaque pays, et l'année où cette population maximale a été enregistrée.

```
ation %>%
```

Résumer les données regroupées

Les résumés regroupés sont également difficiles à réaliser sur des données au format large. Par exemple, en considérant à nouveau le jeu de données `deces_enfants_long`, si vous voulez demander : **Pour chaque pays, quel était le nombre moyen de décès de nourrissons et l'écart-type (variation) des décès ?**

Avec des données au format long, c'est simple :

```
nfants_long %>%
_by(pays) %>%
ize(moyen_deces = mean(nombre_deces),
    sd_deces = sd(nombre_deces))
```

```
## # A tibble: 5 × 3
##   pays                 moyen_deces sd_deces
##   <chr>                      <dbl>    <dbl>
## 1 Afghanistan              68450   4466.
## 2 Albania                    350.    45.2
## 3 Algeria                  21033.   484.
## 4 Angola                   72767.  4513.
## 5 Antigua and Barbuda         10.7    0.816
```

Avec des données au format large, par contre, trouver la moyenne est moins intuitif…

```
nfants_large %>%
se() %>%
(moyen_deces = sum(x2010, x2011, x2012,
                   x2013, x2014, x2015, na.rm = T)/6)
```

```
## # A tibble: 5 × 8
## # Rowwise:
##   pays                 x2010 x2011 x2012 x2013 x2014 x2015
##   <chr>                <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Afghanistan          74600 72000 69500 67100 64800 62700
## 2 Angola               79100 76400 73700 71200 69000 67200
```

```
## 3 Albania                 420   384   354   331   313   301
## 4 United Arab Emirates    683   687   686   681   672   658
## 5 Argentina              9550  9230  8860  8480  8100  7720
## # ℹ 1 more variable: moyen_deces <dbl>
```

Et trouver l'écart-type serait très difficile. (Nous ne pouvons penser à aucune façon d'y parvenir, en fait.)

Pratique 5

Pour cette question de pratique, vous travaillerez à nouveau avec le jeu de données `population` en format long du package `tidyr`.

Utilisez `group_by()` et `summarize()` pour obtenir, pour chaque pays, la population maximale signalée, la population minimale signalée, et la population moyenne signalée sur les années disponibles dans les données. Vos données devraient avoir quatre colonnes, "country", "max_population", "min_population" et "moyen_population".

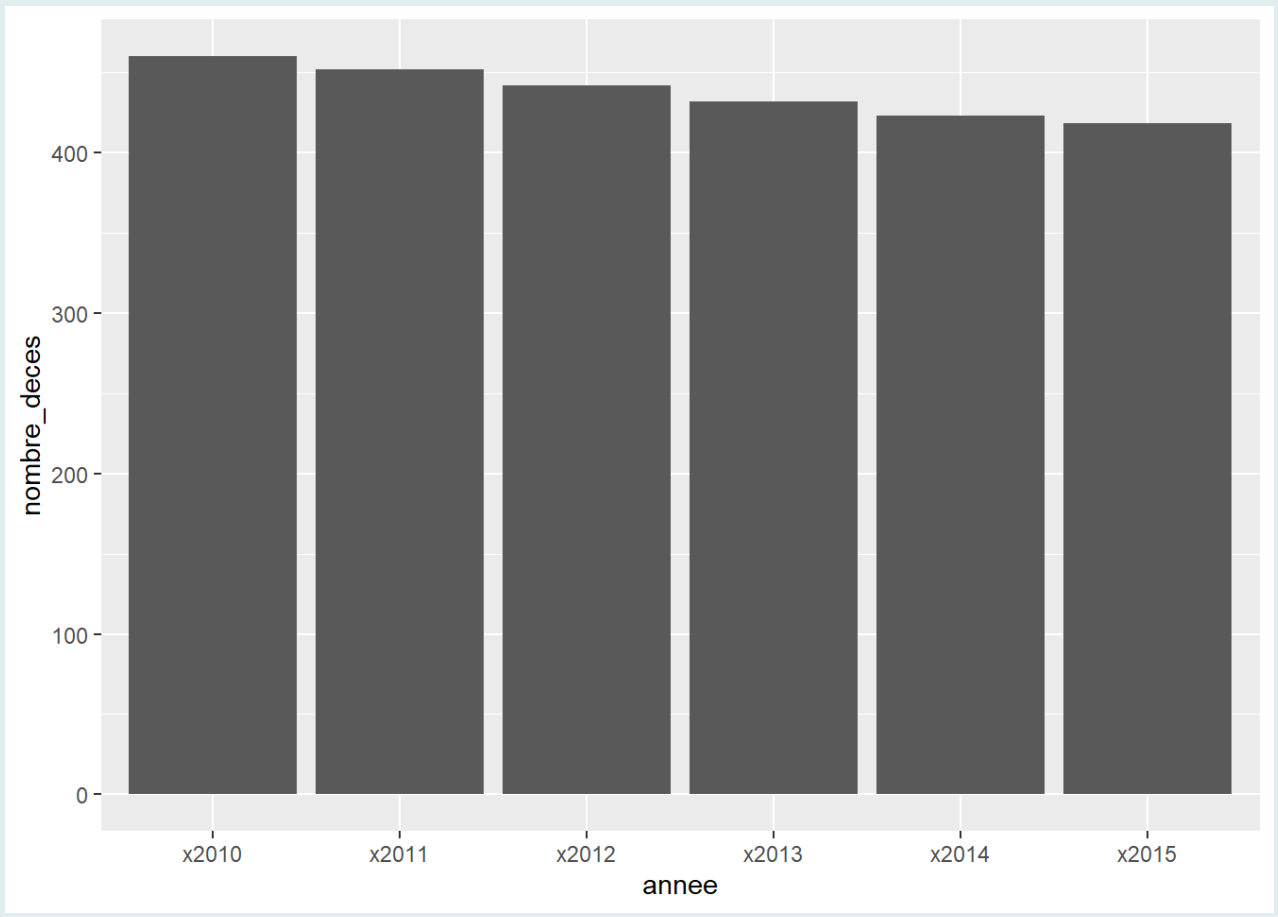
```
ation %>%
```

Conception des graphiques

Enfin, l'une des tâches d'analyse de données qui est LE PLUS entravée par les formats larges est la conception de graphiques. Vous n'avez peut-être pas encore de connaissance préalable de {ggplot} et de comment concevoir des graphiques, nous verrons donc les figures sans approfondir le code. Ce que vous devez retenir est : de nombreux graphiques avec ggplot sont également possibles uniquement avec des données au format long.

Considérez à nouveau les données sur les décès de nourrissons `deces_enfants_long`. Nous allons représenter le nombre de décès pour la Belgique par année :

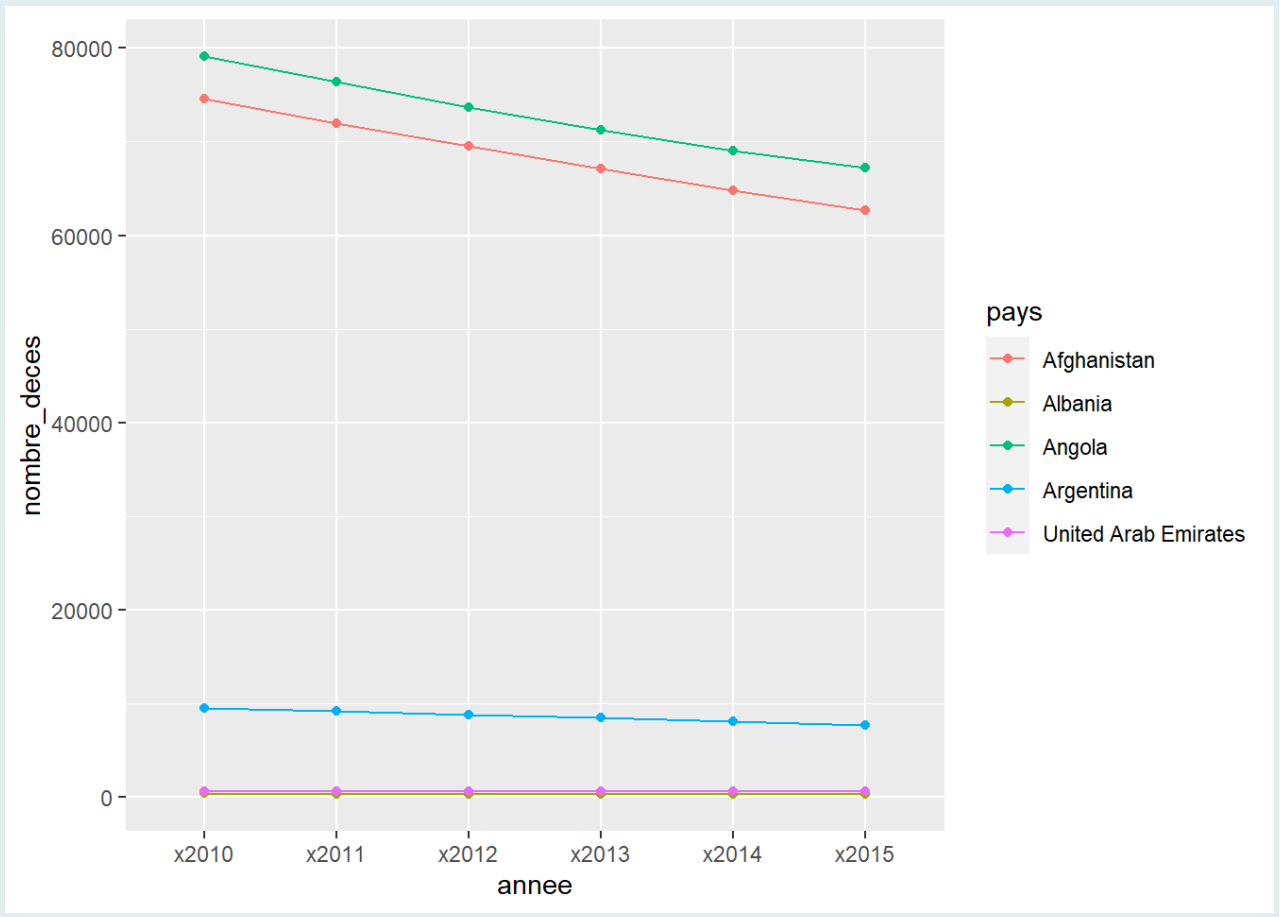
```
nfants_long %>%
(pays == "Belgium") %>%
() +
col(aes(x = annee, y = nombre_deces))
```



La conception du graphique fonctionne parce que nous pouvons donner la variable `annee` pour l'axe des x. Dans le format long, `annee` est une variable à part entière. Dans le format large, il n'y aurait pas une telle variable à passer à l'axe des x.

Un autre graphique qui ne serait pas possible sans un format long :

```
nfants_long %>%
30) %>%
(aes(x = annee, y = nombre_deces, group = pays, color = pays)) +
line() +
point()
```



Encore une fois, la raison est la même, nous devons indiquer au graphique ce qu'il faut utiliser comme axe des x et des y et il est nécessaire d'avoir ces variables dans leurs propres colonnes (comme organisé dans le format long).

Le pivot peut être difficile

Nous avons principalement examiné ici des exemples très simples de pivot, mais dans la pratique, le pivot peut être très difficile à réaliser avec précision. C'est parce que les données avec lesquelles vous travaillez peuvent ne pas avoir toutes les informations nécessaires pour un pivot réussi, ou les données peuvent contenir des erreurs qui vous empêchent de pivoter correctement.

Lorsque vous rencontrez de tels cas, nous vous recommandons de consulter la documentation officielle du pivot de l'équipe `tidyr`, car elle est assez riche en exemples. Vous pourriez également poster vos questions sur le pivot sur des forums comme Stack Overflow.

Conclusion !

Vous avez maintenant exploré différents jeux de données et comment ils sont soit en format long, soit en format large. En fin de compte, il s'agit simplement de la façon dont vous présentez l'information. Parfois, un format sera plus pratique, et d'autres fois un autre pourrait être le meilleur. Maintenant, vous n'êtes plus limité par le format de vos données : ça ne vous plaît pas ? changez-le !

Contributeurs

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

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement

LAURE VANCAUWENBERGHE

Data analyst, the GRAPH Network
A firm believer in science for good, striving to ally programming, health and education

CAMILLE BEATRICE VALERA

Project Manager and Scientific Collaborator, The GRAPH Network

Solutions des exercices pratiques

Solution exercice pratique 1

"large"

Solution exercice pratique 2

```
ces_euro_large %>%
_longer(2:8,
        names_to = "annee",
        values_to = "nombre_de_naissances")
```

```
## # A tibble: 5 × 3
##   pays    annee nombre_de_naissances
##   <chr>   <chr>                <dbl>
## 1 Belgium x2015               122274
## 2 Belgium x2016               121896
## 3 Belgium x2017               119690
## 4 Belgium x2018               118319
## 5 Belgium x2019               117695
```

Solution exercice pratique 3

```
population %>%
_vot_wider(names_from = year,
           values_from = population)
```

```
## # A tibble: 5 × 20
##   country          `1995` `1996` `1997` `1998` `1999` `2000`
##   <chr>             <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Afghanistan    17586073 1.84e7 1.90e7 1.95e7 2.00e7 2.06e7
## 2 Albania         3357858 3.34e6 3.33e6 3.33e6 3.32e6 3.30e6
## 3 Algeria        29315463 2.98e7 3.03e7 3.08e7 3.13e7 3.17e7
## 4 American Samoa    52874 5.39e4 5.49e4 5.59e4 5.68e4 5.75e4
## 5 Andorra           63854 6.43e4 6.41e4 6.38e4 6.41e4 6.54e4
## # ℹ 13 more variables: `2001` <dbl>, `2002` <dbl>,
## #   `2003` <dbl>, `2004` <dbl>, `2005` <dbl>, …
```

Solution exercice pratique 4

```
population %>%
up_by(country) %>%
ter(population == max(population)) %>%
roup()
```

```
## # A tibble: 5 × 3
##   country         year population
##   <chr>          <dbl>      <dbl>
## 1 Afghanistan     2013   30551674
## 2 Albania         1995    3357858
## 3 Algeria         2013   39208194
## 4 American Samoa  2004      59262
## 5 Andorra         2006      81877
```

Solution exercice pratique 5

```r
on %>%
by(country) %>%
ise(max_population = max(population),
    min_population = min(population),
    moyen_population = mean(population))
```

```
## # A tibble: 5 × 4
##   country        max_population min_population
##   <chr>                   <dbl>          <dbl>
## 1 Afghanistan          30551674       17586073
## 2 Albania               3357858        3150143
## 3 Algeria              39208194       29315463
## 4 American Samoa          59262          52874
## 5 Andorra                 81877          63799
## # ℹ 1 more variable: moyen_population <dbl>
```