Manipuler des bases de données avec `dplyr`

À votre tour #0: chargez les données

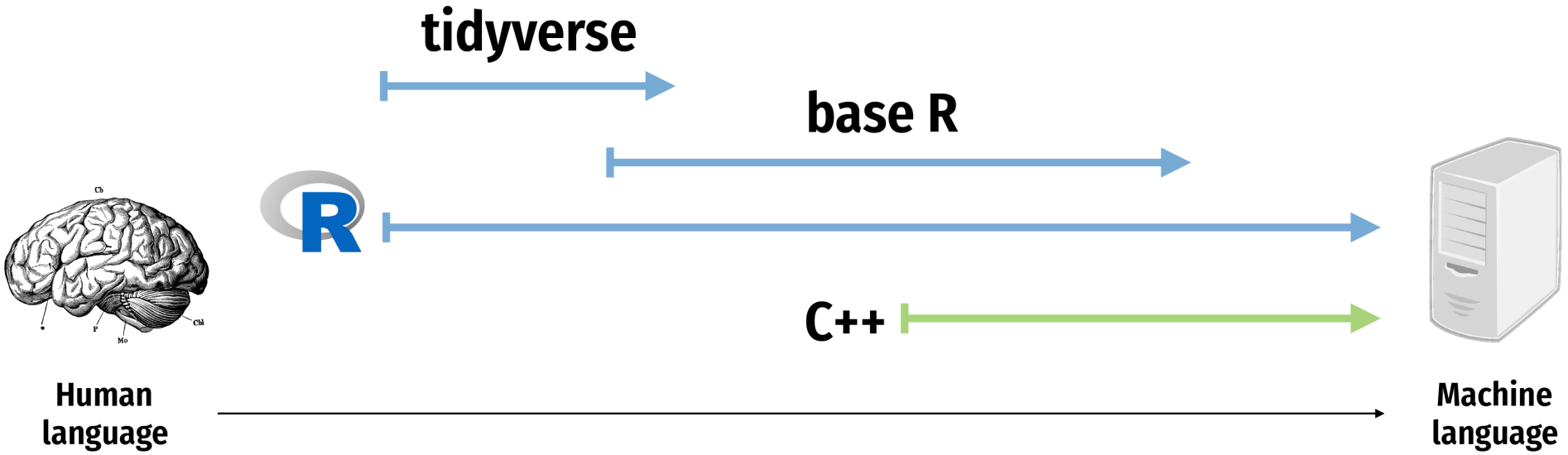
1. Compilez le bloc `setup`
2. Jetez un coup d'oeil aux données `gapminder`

02:00

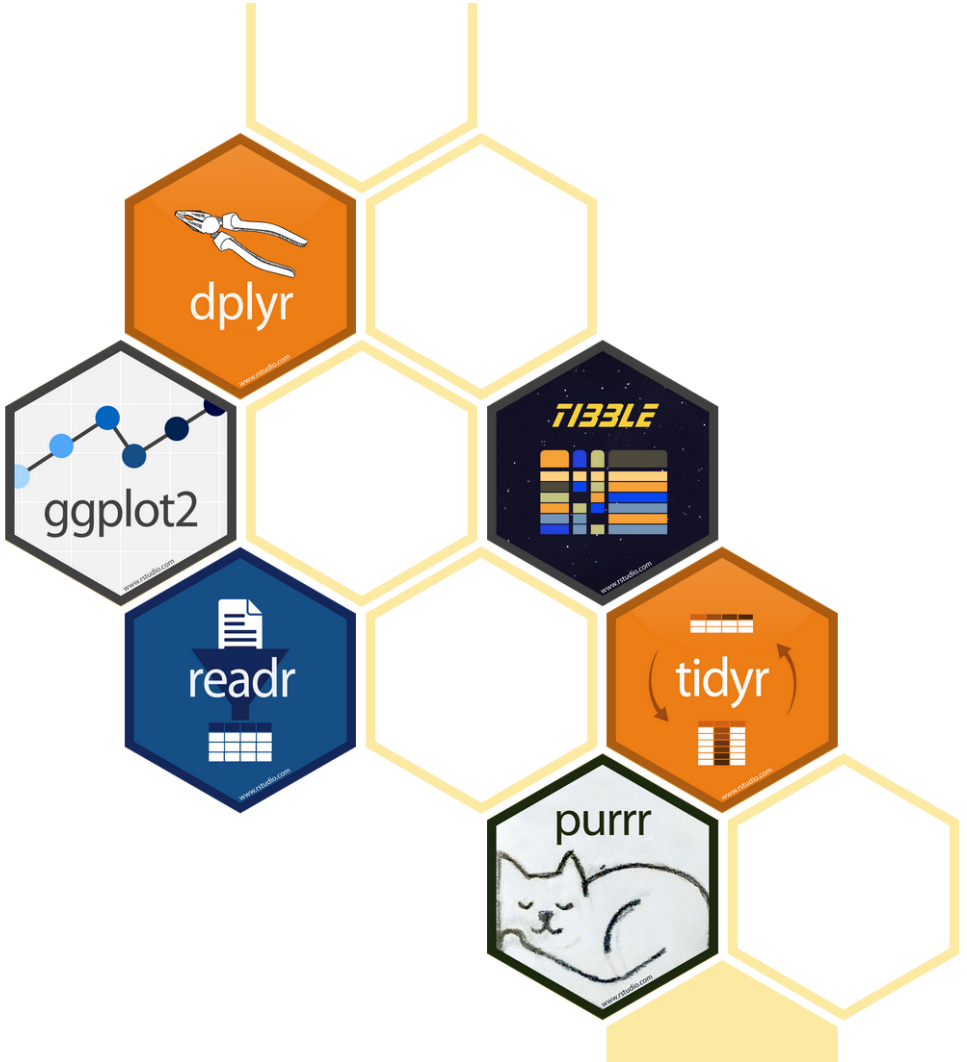
```
gapminder
```

```
## # A tibble: 1,704 × 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # … with 1,694 more rows
```

tidyverse

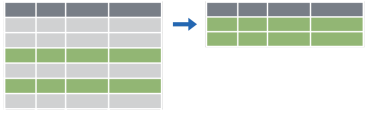


tidyverse



`dplyr`: verbes pour manipuler des données

Extraire des lignes avec `filter()`

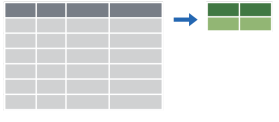


Extraire des colonnes avec `select()`

Arranger/trier les lignes avec `arrange()`

Créer/modifier des colonnes avec `mutate()`

Résumer des sous-ensembles avec `group_by() |> summarize()`



filter()

filter()

Extraire des lignes selon une condition logique

```
filter(.data = DATA, ...)
```

- **DATA** = tableau de données à transformer
- **...** = Un ou plusieurs tests

`filter()` retourne chaque ligne pour lequel le test retourne `TRUE`

```
filter(.data = gapminder, country == "Denmark")
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
...	...	...

country	continent	year
Denmark	Europe	1952
Denmark	Europe	1957
Denmark	Europe	1962
Denmark	Europe	1967
Denmark	Europe	1972
Denmark	Europe	1977

filter()

```
filter(.data = gapminder,
       country == "Denmark")
```

Un signe =
pour l'assignation d'un argument

Deux signes ==
teste l'égalité entre objets
retourne TRUE ou FALSE)

Tests logiques

Teste	Signification	Test	Signification
`x < y`	plus petit que	`x %in% y`	dans (élément d'un vecteur)
`x > y`	plus grand que	`is.na(x)`	valeur manquante?
`==`	égale	`!is.na(x)`	valeurs non manquantes
`x <= y`	plus petit ou égal à		
`x >= y`	plus grand ou égal à		
`x != y`	différent de		

À votre tour #1: filtrer

Utilisez `filter()` et des tests logiques pour montrer...

1. Les données du Canada
2. Toutes les données de pays situés en Océanie (Oceania)
3. Lignes pour l'espérance de vie supérieur à 82 ans

03:00

```
filter(gapminder, country == "Canada")
```

```
filter(gapminder, continent == "Oceania")
```

```
filter(gapminder, lifeExp > 82)
```

Erreur fréquente

Utiliser = plutôt que ==

```
filter(gapminder,
       country = "Canada")
```

```
filter(gapminder,
       country == "Canada")
```

Mettre entre accolade

```
filter(gapminder,
       country == Canada)
```

```
filter(gapminder,
       country == "Canada")
```

`filter()` avec des conditions multiples

Extrait les lignes qui valident *tous* les énoncés

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark", year > 2000)
```

country	continent	year
Afghanistan	Asia	1952
Afghanistan	Asia	1957
Afghanistan	Asia	1962
Afghanistan	Asia	1967
Afghanistan	Asia	1972
...	...	...

country	continent	year
Denmark	Europe	2002
Denmark	Europe	2007

Opérateurs logiques

Opérateur	Signification
`a & b`	et
`a | b`	ou
`!a`	pas

Par défaut, union ("et")

Commandes équivalentes

```
filter(gapminder, country == "Denmark", year > 2000)
```

```
filter(gapminder, country == "Denmark" & year > 2000)
```

À votre tour #2: filtrer

Utilisez `filter()` et les opérateurs logiques pour montrer...

1. Les données du Canada avant 1970
2. Les pays où l'espérance de vie en 2007 est inférieure à 50
3. Les pays hors d'Afrique où l'espérance de vie en 2007 est inférieur à 50

04:00

```
filter(gapminder, country == "Canada", year < 1970)
```

```
filter(gapminder, year == 2007, lifeExp < 50)
```

```
filter(gapminder, year == 2007, lifeExp < 50,
       continent != "Africa")
```

Erreurs fréquentes

Rassembler plusieurs conditions en une seule

```
filter(gapminder, 1960 < year < 1980)
```

```
filter(gapminder,
       year > 1960, year < 1980)
```

Utiliser plusieurs tests plutôt que `%in%`

```
filter(gapminder,
       country == "Mexico",
       country == "Canada",
       country == "United States")
```

```
filter(gapminder,
       country %in% c("Mexico", "Canada",
                      "United States"))
```

Syntaxe commune

Chaque verbe de `dplyr` a la même syntaxe

Base de données comme premier argument, retourne une base de données

```
VERB(DATA, ...)
```

- **`VERB`** = fonction `dplyr`/verbe
- **`DATA`** = base de données à transformer
- **`...`** = commandes pour le verbe

mutate()

Créer de nouvelles colonnes

```
mutate(.data, ...)
```

- **`DATA`** = base de données à transformer
- **`...`** = colonnes à modifier/créer

```
mutate(gapminder, gdp = gdpPercap * pop)
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp
Afghanistan	1952	...	6567086330
Afghanistan	1957	...	7585448670
Afghanistan	1962	...	8758855797
Afghanistan	1967	...	9648014150
Afghanistan	1972	...	9678553274
Afghanistan	1977	...	11697659231

```
mutate(gapminder, gdp = gdpPercap * pop,
                  pop_mil = round(pop / 1000000))
```

country	year	gdpPercap	pop
Afghanistan	1952	779.4453145	8425333
Afghanistan	1957	820.8530296	9240934
Afghanistan	1962	853.10071	10267083
Afghanistan	1967	836.1971382	11537966
Afghanistan	1972	739.9811058	13079460
...	...	...	...

country	year	...	gdp	pop_mil
Afghanistan	1952	...	6567086330	8
Afghanistan	1957	...	7585448670	9
Afghanistan	1962	...	8758855797	10
Afghanistan	1967	...	9648014150	12
Afghanistan	1972	...	9678553274	13
Afghanistan	1977	...	11697659231	15

ifelse()

Effectuer des tests conditionnels

```
ifelse(TEST,
       VALEUR_SI_TRUE,
       VALEUR_SI_FALSE)
```

- **TEST** = un test logique
- **VALEUR_SI_TRUE** = valeur si vrai
- **VALUE_IF_FALSE** = valeur si faux

```
mutate(gapminder,
       after_1960 = year > 1960)
```

```
mutate(gapminder,
       after_1960 = ifelse(year > 1960,
                           "After 1960",
                           "Before 1960"))
```

À votre tour #3: transformer

Utilisez `mutate()` pour ajouter les colonnes...

1. `africa`, qui est vrai (`TRUE`) si le pays est situé sur le continent africain
2. `log_gdpPercap` pour le log PIB par capita (indice: utiliser `log()`)
3. `africa_asia` avec comme valeur `"Afrique ou Asie"` si le pays est dans un des deux continents, sinon `"Autre continent"`

05:00

```
mutate(gapminder,
       africa = ifelse(continent == "Africa",
                                    TRUE, FALSE))
```

```
mutate(gapminder,
       log_gdpPercap = log(gdpPercap))
```

```
mutate(gapminder,
       africa_asia =
         ifelse(continent %in% c("Africa", "Asia"),
                "Afrique ou Asie",
                "Autre continent"))
```

Comment procéder avec plusieurs verbes?

Créer un jeu de données pour 2002 *et* calculer le log PIB par capita

Solution 1: variables auxiliaires

```
gapminder_2002 <- filter(gapminder, year == 2002)

gapminder_2002_log <- mutate(gapminder_2002,
                             log_gdpPercap = log(gdpPercap))
```

Comment procéder avec plusieurs verbes?

Créer un jeu de données pour 2002 *et* calculer le log PIB par capita

Solution 2: fonctions emboîtées

```
filter(mutate(gapminder_2002,
              log_gdpPercap = log(gdpPercap)),
       year == 2002)
```

Comment procéder avec plusieurs verbes?

Créer un jeu de données pour 2002 *et* calculer le log PIB par capita

Solution 3: tuyaux!

L'opérateur tuyau, `|>` prend un objet à gauche et l'assigne au premier argument de la fonction de droite

```
gapminder |> filter(_, country == "Canada")
```

Comment procéder avec plusieurs verbes?

Ces deux lignes donnent le même résultat

```
filter(gapminder, country == "Canada")
```

```
gapminder |> filter(country == "Canada")
```

Comment procéder avec plusieurs verbes?

Créer un jeu de données pour 2002 *et* calculer le log PIB par capita

Solution 3: Tuyaux!

```
gapminder |>
  filter(year == 2002) |>
  mutate(log_gdpPercap = log(gdpPercap))
```

Améliorer la lisibilité du code avec |>

```
leave_house(get_dressed(get_out_of_bed(wake_up(me, time = "8:00"),
side = "correct"), pants = TRUE, shirt = TRUE), car = TRUE, bike =
FALSE)
```

```
me |>
  wake_up(time = "8:00") |>
  get_out_of_bed(side = "correct") |>
  get_dressed(pants = TRUE, shirt = TRUE) |>
  leave_house(car = TRUE, bike = FALSE)
```

summarize()

Créer un tableau résumé

```
gapminder |> summarize(mean_life = mean(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
…	…	…	…

mean_life
59.47444

summarize()

```
gapminder |> summarize(mean_life = mean(lifeExp),
                       min_life = min(lifeExp))
```

country	continent	year	lifeExp
Afghanistan	Asia	1952	28.801
Afghanistan	Asia	1957	30.332
Afghanistan	Asia	1962	31.997
Afghanistan	Asia	1967	34.02
Afghanistan	Asia	1972	36.088
…	…	…	…

mean_life	min_life
59.47444	23.599

À votre tour #4: résumer

Utilisez `summarize()` pour calculer...

1. La première année des mesures (minimum)
2. La dernière année des mesures (maximum)
3. Le nombre de lignes dans la base de données (utilisez l'aide mémoire)
4. Le nombre de pays distincts dans la base de données (utilisez l'aide mémoire)

04:00

```
gapminder |>
  summarize(first = min(year),
            last = max(year),
            num_rows = n(),
            num_unique = n_distinct(country))
```

first	last	num_rows	num_unique
1952	2007	1704	142

À votre tour #5: résumer

Utilisez `filter()` et `summarize()` pour calculer

1. le nombre de pays
2. l'espérance de vie médiane

pour le continent africain en 2007.

04:00

```
gapminder |>
  filter(continent == "Africa", year == 2007) |>
  summarise(n_countries = n_distinct(country),
            med_le = median(lifeExp))
```

n_countries	med_le
52	52.9265

group_by()

Assembler les lignes en groupes selon les valeurs d'une colonne

```
gapminder |> group_by(continent)
```

Rien n'apparaît!

Outil puissant si combiné avec `summarize()`

```
gapminder |>
  group_by(continent) |>
  summarize(n_countries = n_distinct(country))
```

continent	n_countries
Africa	52
Americas	25
Asia	33
Europe	30
Oceania	2

```
pollution |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

mean	sum	n
42	252	6

```
pollution |>
  group_by(city) |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

city	mean	sum	n
Beijing	88.5	177	2
London	19.0	38	2
New York	18.5	37	2

```
pollution |>
  group_by(particle_size) |>
  summarize(mean = mean(amount), sum = sum(amount), n = n())
```

city	particle_size	amount
New York	Large	23
New York	Small	14
London	Large	22
London	Small	16
Beijing	Large	121
Beijing	Small	56

particle_size	mean	sum	n
Large	55.33333	166	3
Small	28.66667	86	3

À votre tour #6: grouper et résumer

Trouvez l'espérance de vie minimum, maximum et médiane par continent

Trouvez l'espérance de vie minimum, maximum et médiane par continent pour 2007 uniquement

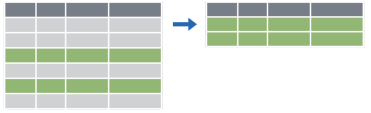
05:00

```
gapminder |>
  group_by(continent) |>
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

```
gapminder |>
  filter(year == 2007) |>
  group_by(continent) |>
  summarize(min_le = min(lifeExp),
            max_le = max(lifeExp),
            med_le = median(lifeExp))
```

`dplyr`: verbes pour manipuler des données

Extraire des lignes avec `filter()`



Extraire des colonnes avec `select()`

Arranger/trier les lignes avec `arrange()`

Créer/modifier des colonnes avec `mutate()`

Résumer des sous-ensembles avec `group_by() |> summarize()`