Visualiser et manipuler les données avec 

Anicet Ebou

@Ebedthan
@anicetebou

bit.ly/ae-slides-satRday

satRday Abidjan 2020

Partie 1

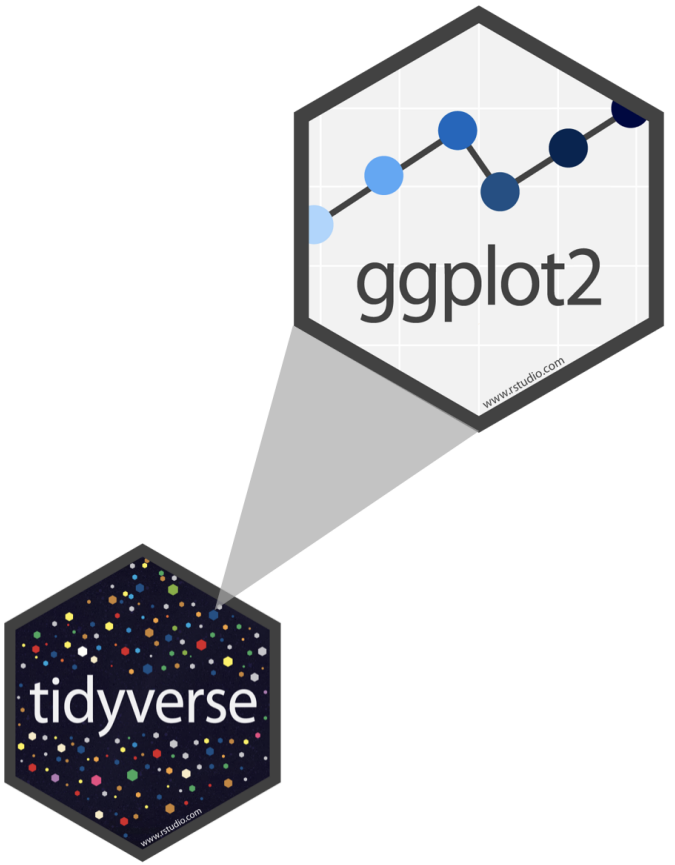
La visualisation des données

Qu'est ce que c'est?

> "The simple graph has brought more information to the data analyst's mind than any other device." — John Tukey

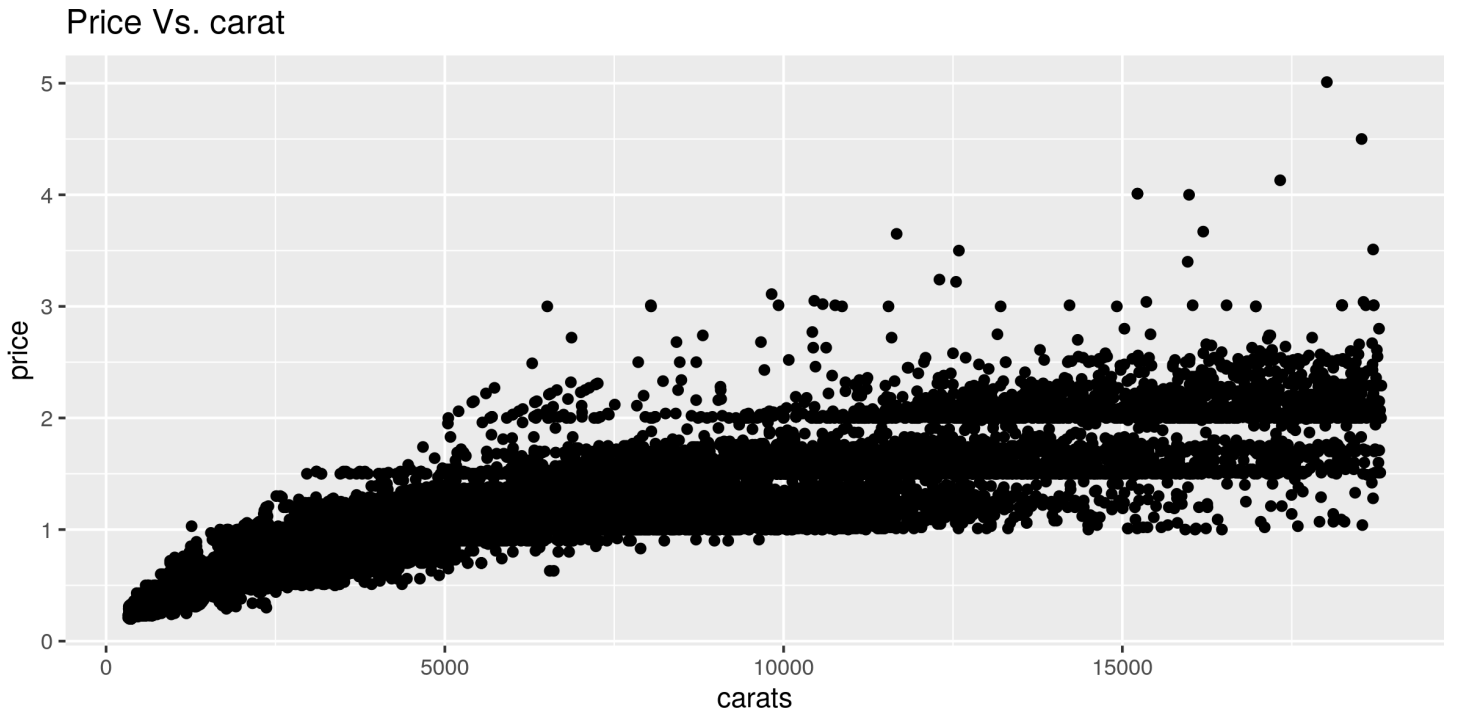
- La visualisation des données est la création et l'étude de la représentation visuelle des données.
- Il existe de nombreux outils pour visualiser les données (R est l'un d'eux), et de nombreuses approches/systèmes dans R pour faire des visualisations de données (**ggplot2** est l'un d'eux, et c'est ce que nous allons utiliser).

ggplot2 $\in$ tidyverse



- ggplot2 est le package de visualisation de données de tidyverse.
- Le **gg** dans ggplot2 signifie grammar of graphics (grammaire des graphiques)
- La grammaire des graphiques permet de décrire (et donc de construire) de facon précise les composants d'un graphique.

```
ggplot(data = diamonds, mapping = aes(x = price, y = carat)) +
  geom_point() +
  labs(title = "Price Vs. carat",
       x = "carats", y = "price")
```



Quelles sont les fonctions pour dessiner ce graphique? Quelles sont les données utilisées pour le graphique? Quelles sont les variables en abscisse et en ordonnée?

Bonjour ggplot2!

- `ggplot()` est la fonction principale du package ggplot2 et les graphiques sont construits **en couches**.
- La structure du code des graphiques peut être synthétisé comme:

```
ggplot +
  geom_xxx
```

ou plus précisement

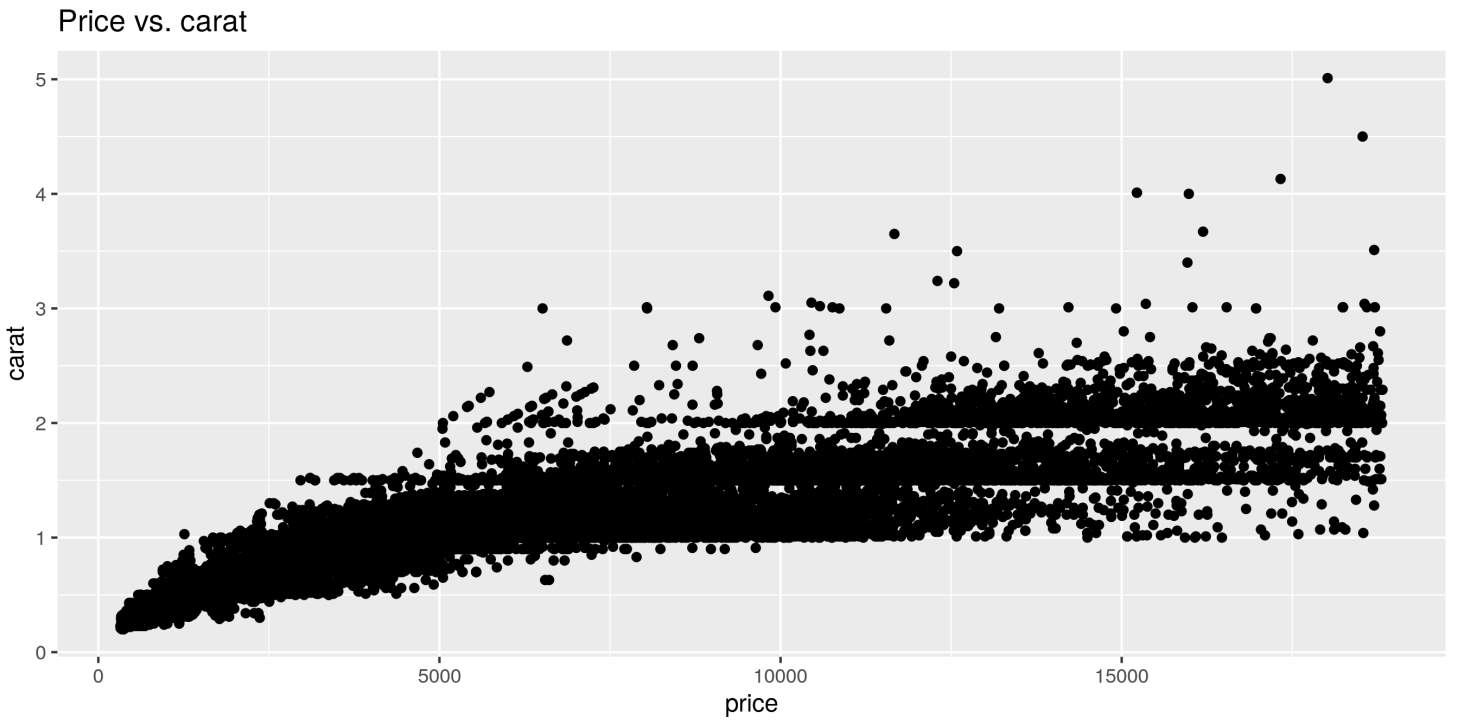
```
ggplot(data = [dataset], mapping = aes(x = [x-variable], y = [y=variable])) +
  geom_xxx() +
  other_options
```

- Il faut charger le package ggplot2 avant de pouvoir utiliser la fonction ggplot()

```
library(ggplot2)
```

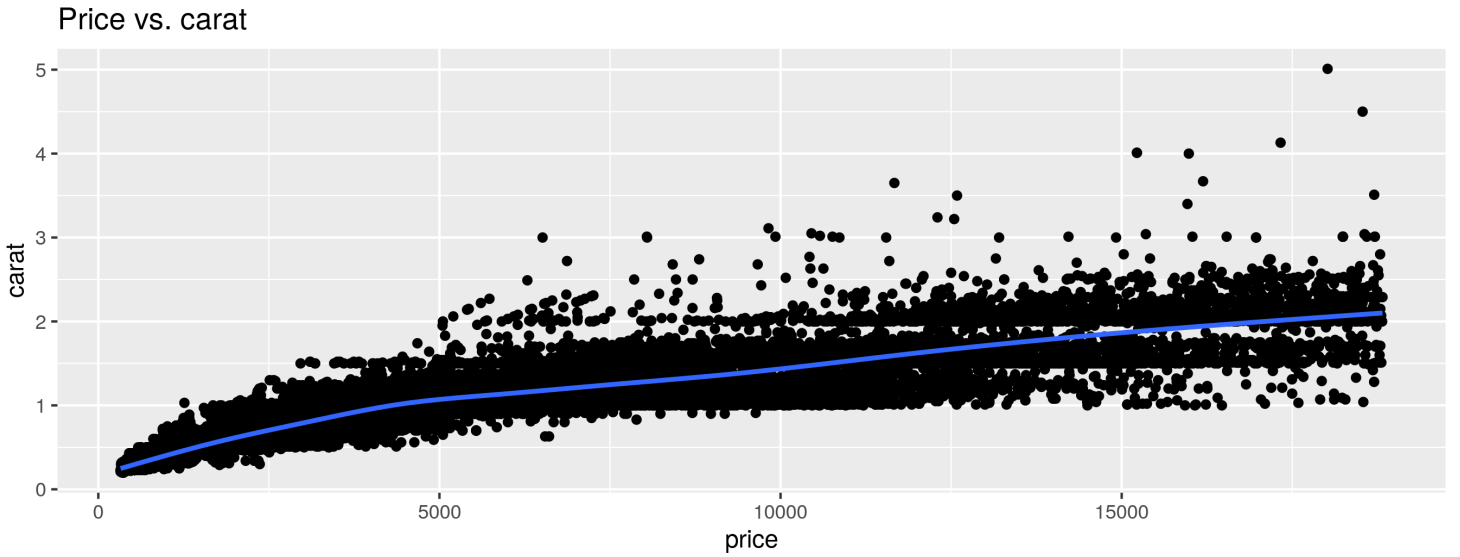
Price vs Carat

```
ggplot(data = diamonds, mapping = aes(x = price, y = carat)) +
  geom_point() +
  labs(title = "Price vs. carat",
       x = "price", y = "carat")
```



```
ggplot(data = diamonds, mapping = aes(x = price, y = carat)) +
  geom_point() +
  geom_smooth() +
  labs(title = "Price vs. carat",
       x = "price", y = "carat")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



> Quelle est la différence entre ce graphe et le précédent? Qu'est ce que geom_smooth() apporte de plus?

Point important

En clair voici comment fonctionne la grammaire des graphiques:

Point important

En clair voici comment fonctionne la grammaire des graphiques:

Variables additionnelles

D'autres variables additionnelles peuvent être ajoutés:

- Esthetique
  - forme
  - couleur
  - taille
  - transparence (alpha)
- facetage: plusieurs graphiques montrant différents sous graphiques

Esthetique

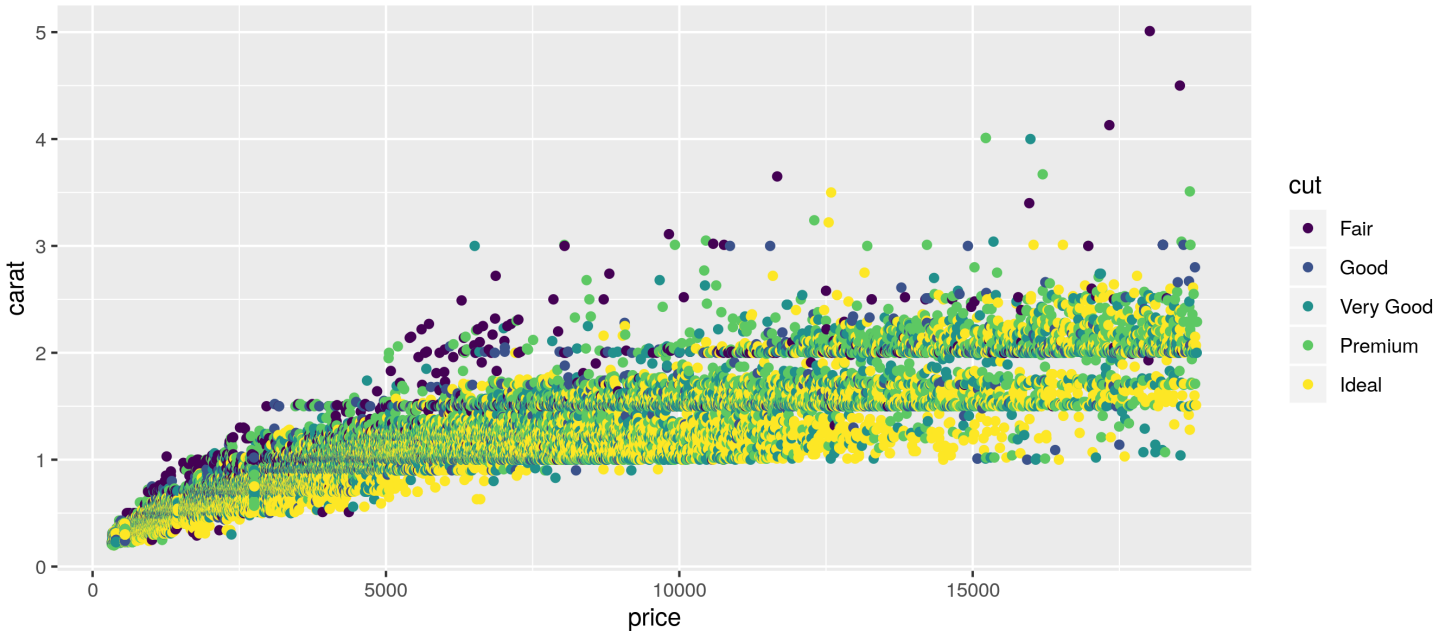
Les options

Les options des graphiques qui peuvent être **liées à des variables** du jeu de données sont

- couleur
- taille
- forme
- transparence (alpha)

Price vs carat + cut

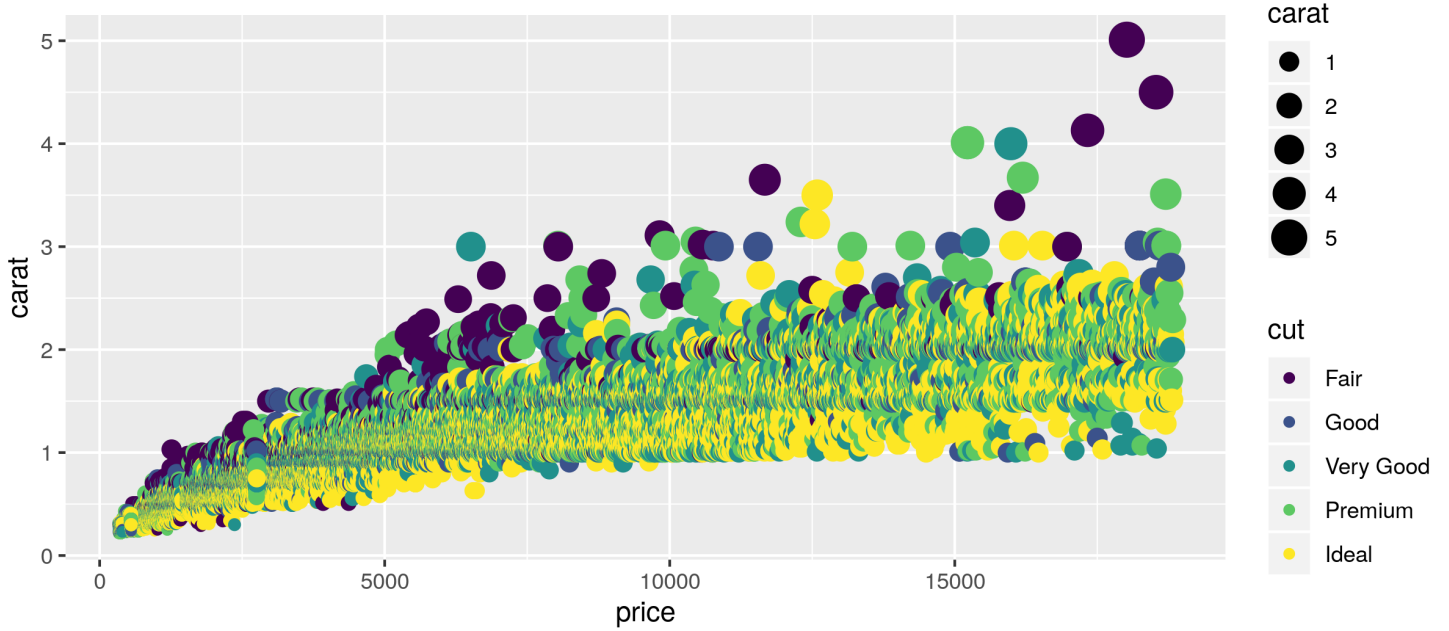
```
ggplot(data = diamonds, mapping = aes(x = price, y = carat,
          color = cut
          )) + geom_point()
```



Price vs carat + cut

Relions la taille (size) au carat (carat):

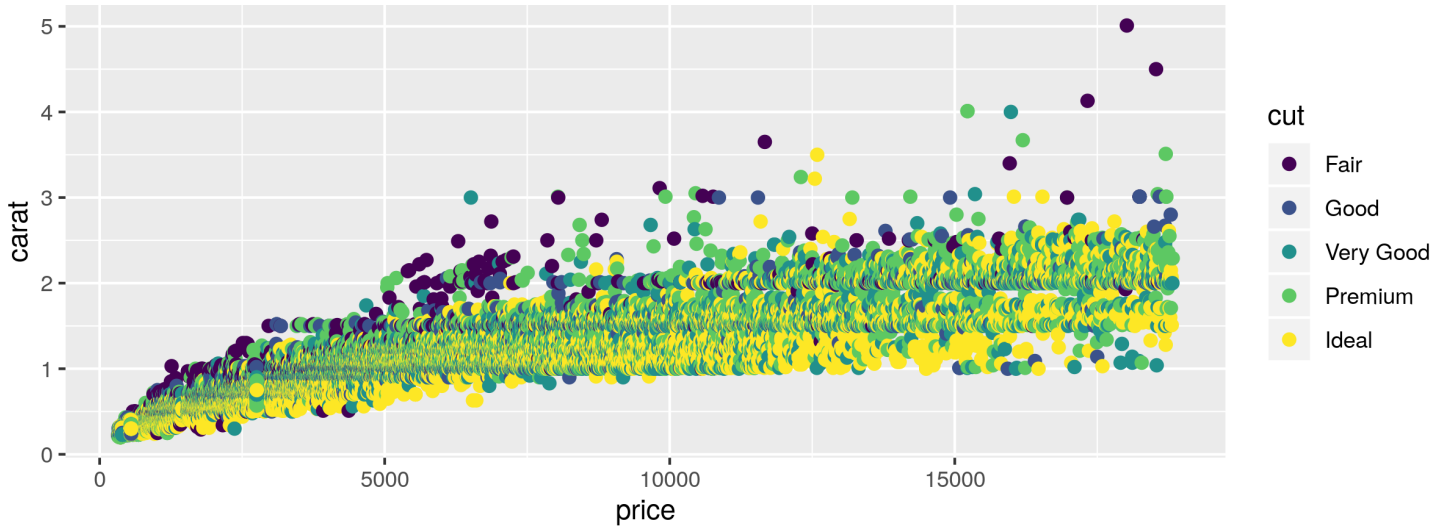
```
ggplot(data = diamonds, mapping = aes(
  x = price, y = carat, color = cut,
  size = carat
  )) + geom_point()
```



Price vs carat + cut

Agrandisons cette fois ci la taille de tout les points *non* sur les valeurs des variables:

```
ggplot(data = diamonds, mapping = aes(
  x = price, y = carat, color = cut)) +
  geom_point(size = 2)
```



Résumons les aesthetics

- Les variables continues sont mesurées sur une échelle continue.
- Les variables dicrètes sont mesurées (ou souvent comptées) sur une échelle discrete.

aesthetics	discrete	continuous
couleur	rainbow of colors	gradient
taille	discrete steps	correspondance lineaire entre le rayon et la valeur
forme	different shape for each	ne doit pas marcher (et ne marche pas)

- Utiliser les aesthetics pour lier les caractéristiques d'un graphique à une variable, définir les caractéristiques **non liées** à des variables à l'intérieur des geom_*().

Histogrammes

```
ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_histogram(binwidth = 1000)
```

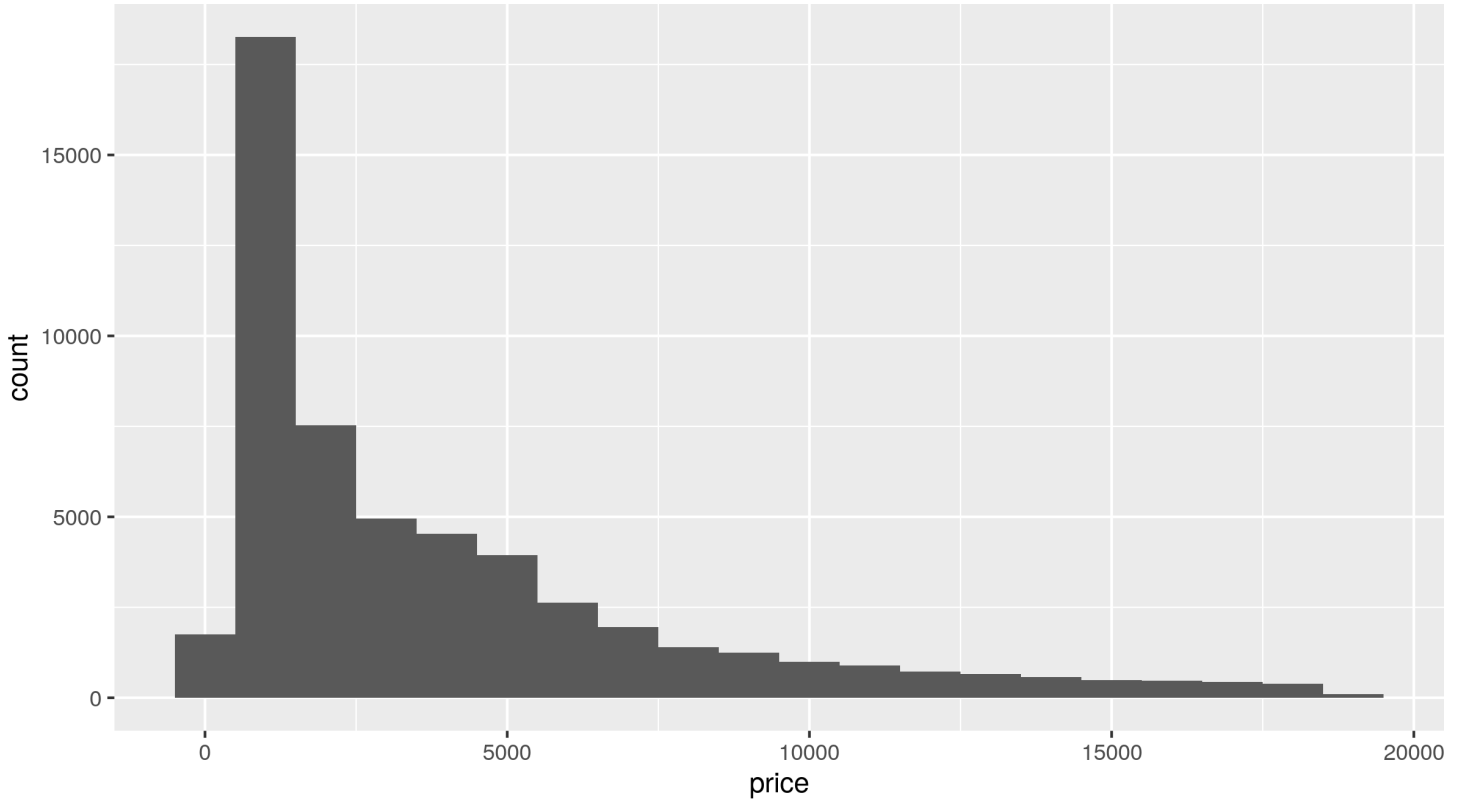
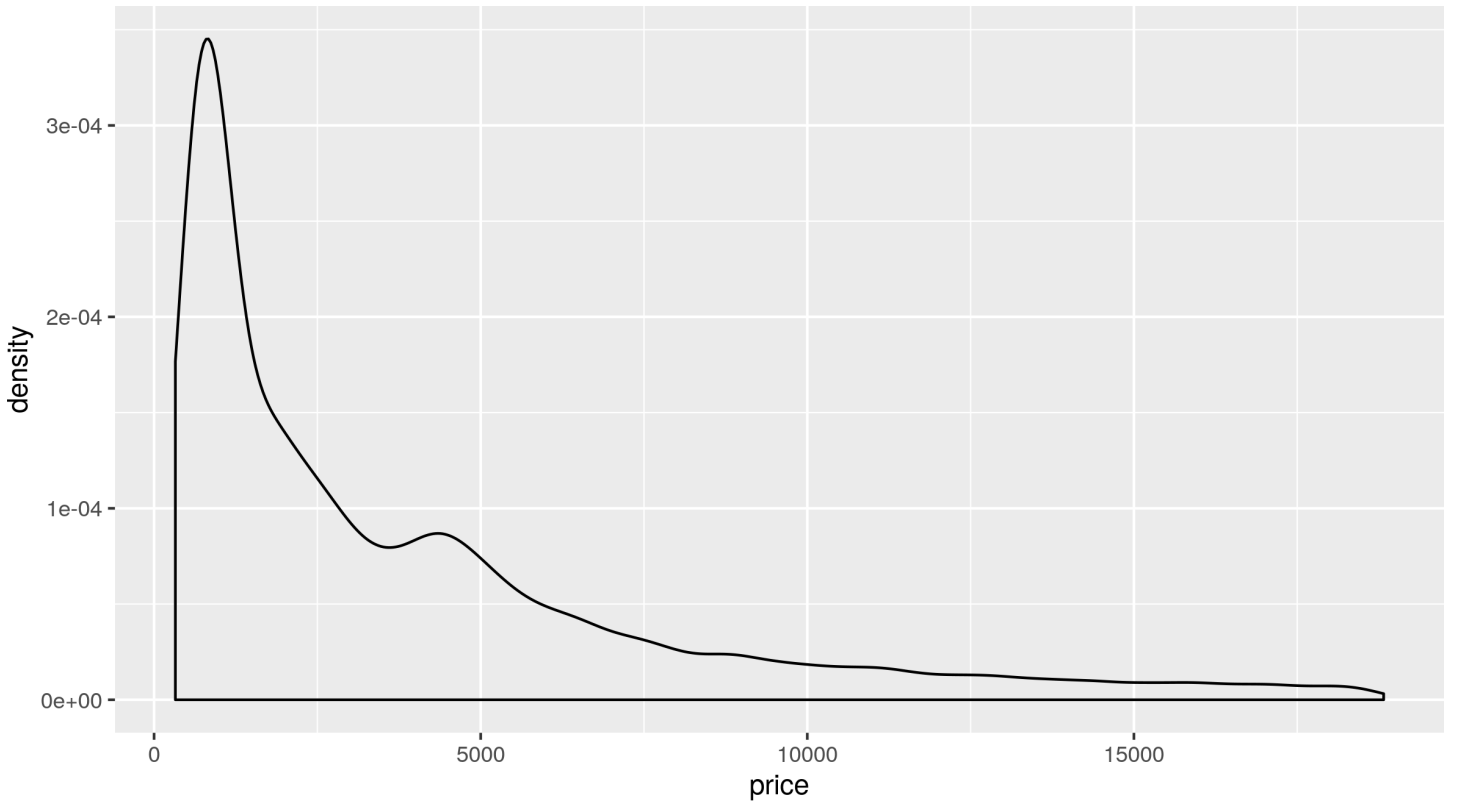


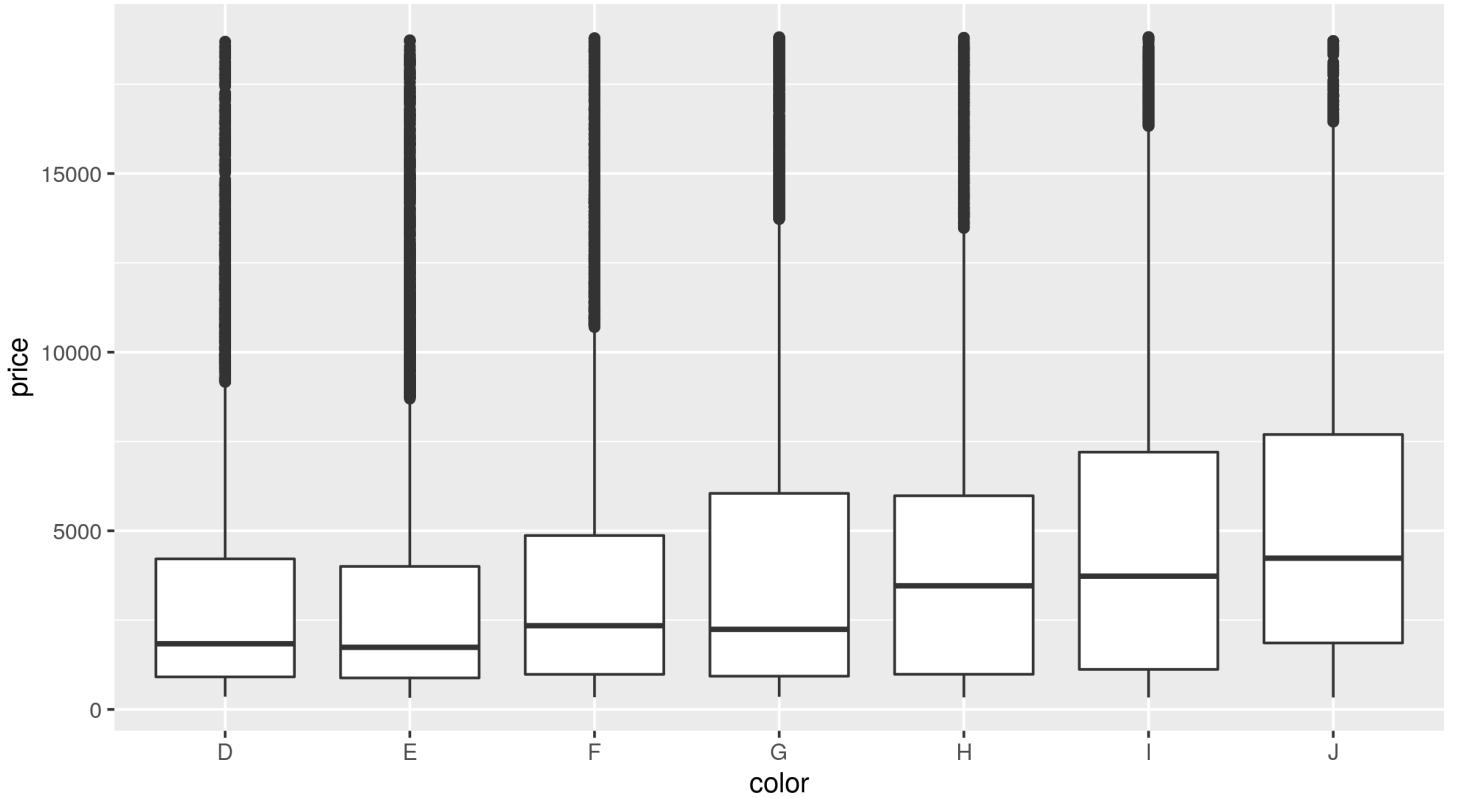
Diagramme de densité

```
ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_density()
```



Boite à moustache

```
ggplot(data = diamonds, mapping = aes(y = price, x = color)) +
  geom_boxplot()
```



Partie 2

La transformation des donnees

Données ordonnées (Tidy data)

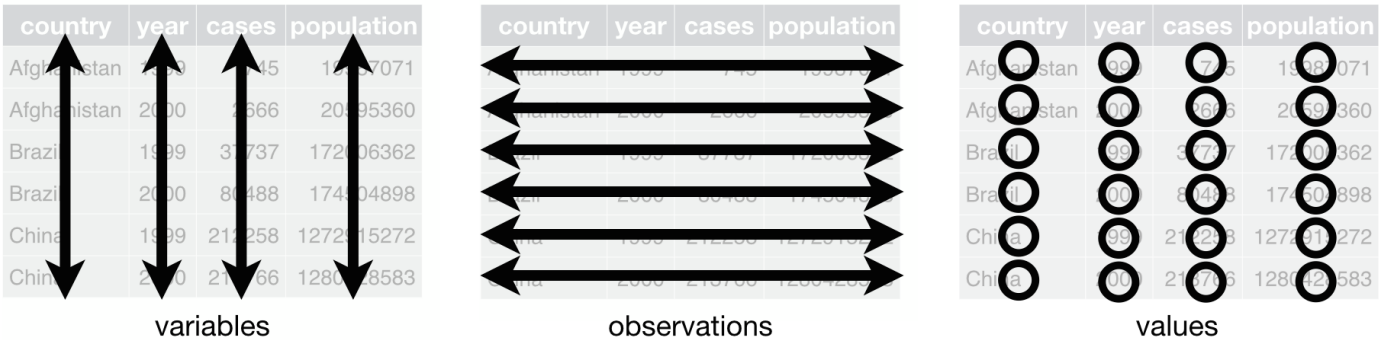
Qu'est ce que c'est?

> "Happy families are all alike; every unhappy family is unhappy is its own way." — Leo Tolstoy

Les règles qui définissent les données ordonnées sont:

- Chaque variable doit avoir sa propre colonne;
- Chaque observation doit avoir sa propre ligne;
- Chaque valeur doit avoir sa propre cellule.

Données ordonnées



R for Data Science - Hadley Wickham, Garret Grolemund

Manipulation des données

Données sur la santé des pays Africains

Le jeu de données est contenue dans le package afrods (une partie des données sur la santé):

```
# devtools::install_github("Ebedthan/afrods")
library(afrods)
health
```

Variables

Pour voir les variables dans un jeu de données:

```
names(health)
```

```
##  [1] "country"
##  [2] "region"
##  [3] "year"
##  [4] "Births attended by skilled health staff (% of total)"
##  [5] "Contraceptive prevalence (% of women ages 15-49)"
##  [6] "One-year-olds immunized with MCV (%)"
##  [7] "Deads kids per couple"
##  [8] "Infant mortality rate"
##  [9] "Under five mortality"
## [10] "Per capita total expenditure on health at average exchange rate (US$)"
## [11] "Estimated HIV Prevalence% - (Ages 15-49)"
## [12] "Life expectancy with projections"
## [13] "Life expectancy"
...
```

Un apercu des données

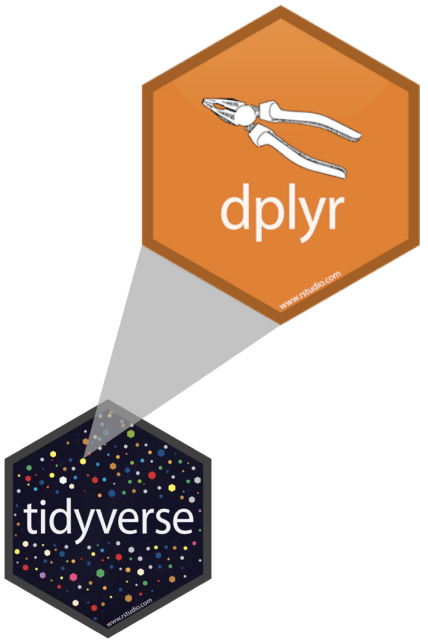
- Dans Rstudio, faire `data(health)`, ensuite cliquer sur le nom du jeu de données dans le panneau environnement.
- Dans la console, utiliser la fonction `glimpse` pour un apercu: `glimpse(health)`.

```
glimpse(health)
```

```
## Observations: 15,208
## Variables: 84
## $ country
## $ region
## $ year
## $ `Births attended by skilled health staff (% of total)`
## $ `Contraceptive prevalence (% of women ages 15-49)`
## $ `One-year-olds immunized with MCV (%)`
## $ `Deads kids per couple`
## $ `Infant mortality rate`
...
```

Une grammaire pour la manipulation des données

dplyr est basé sur des verbes (fonctions) qui facilitent la manipulation des données.



- `filter()`: sélectionne les observations par leurs valeurs.
- `arrange()`: réorganise les lignes.
- `select()`: sélectionne les variables par leurs noms.
- `mutate()`: crée de nouvelles variables en fonction des variables existantes.
- `summarise()`: Résume les valeurs en un seul summary.
- etc.

Une grammaire pour la manipulation des données

Tout les verbes **dplyr** fonctionnent de la même manière:

- Le premier argument est un data frame.

```
verbe(dataframe)
```

- Les arguments des verbes décrivent **quoi faire** avec le data frame, en utilisant **les noms des variables (sans les griffes)**.

```
verbe(dataframe, var1 == value)
```

- Le résultat est un nouveau data frame (Il faut donc sauver ce nouveau data frame).

```
df <- verbe(dataframe, var1 == value)
```

Filtrer les observations avec `filter()`

Pour les données uniquement sur la Côte d'Ivoire:

```
filter(health, country == "Cote d'Ivoire")
```

```
##           country         region year
## 1   Cote d'Ivoire Western Africa 1545
## 2   Cote d'Ivoire Western Africa 1586
## 3   Cote d'Ivoire Western Africa 1590
## 4   Cote d'Ivoire Western Africa 1610
## 5   Cote d'Ivoire Western Africa 1614
## 6   Cote d'Ivoire Western Africa 1635
## 7   Cote d'Ivoire Western Africa 1644
## 8   Cote d'Ivoire Western Africa 1657
## 9   Cote d'Ivoire Western Africa 1660
...
```

Introduisons l'opérateur pipe: %>%

Ecrire

```
filter(health, country == "Cote d'Ivoire")
```

```
##           country         region year
## 1   Cote d'Ivoire Western Africa 1545
## 2   Cote d'Ivoire Western Africa 1586
...
```

Equivaut à (avec la pipe)

```
health %>% filter(country == "Cote d'Ivoire")
```

```
##           country         region year
## 1   Cote d'Ivoire Western Africa 1545
## 2   Cote d'Ivoire Western Africa 1586
...
```

%>% permet d'effectuer des opérations multiples sur les mêmes données plus facilement tout en gardant le code lisible.

Introduisons l'opérateur pipe: %>%

Par exemple

```
cote_ivoire <- filter(health, country == "Cote d'Ivoire")
infant_mortality_ci <- select(cote_ivoire, `Infant mortality rate`)
```

est plus simplement effectué par

```
health %>%
  filter(country == "Cote d'Ivoire") %>%
  select(`Infant mortality rate`)
```

Introduisons l'opérateur pipe: %>%

L'opérateur pipe est implementé dans le package **magrittr** et est prononcé "(et) ensuite".

Filter sur plusieurs conditions à la fois

Pour les données sur la Côte d'Ivoire entre 1960 et 2010

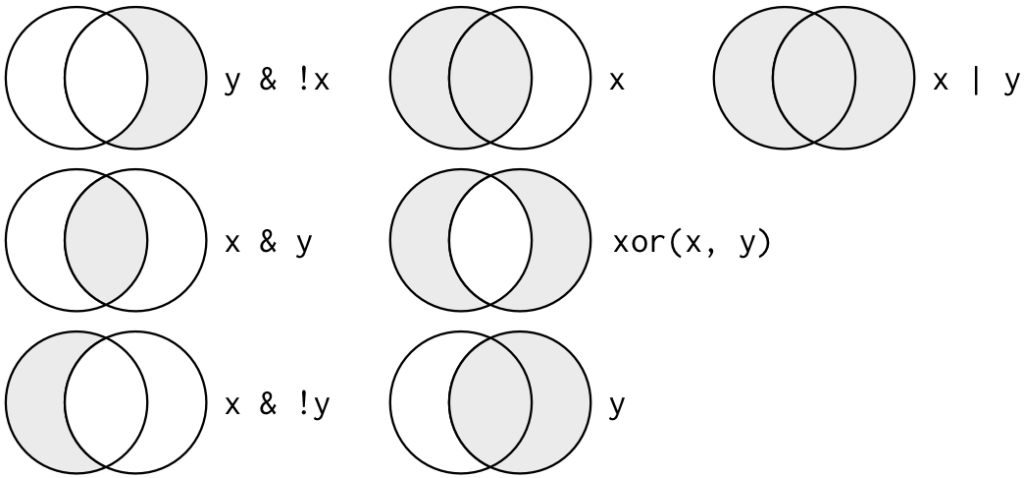
```
health %>%
  filter(country == "Cote d'Ivoire", year %in% c(1960:2010))
```

```
##          country         region year
## 1  Cote d'Ivoire Western Africa 1960
## 2  Cote d'Ivoire Western Africa 1961
## 3  Cote d'Ivoire Western Africa 1962
## 4  Cote d'Ivoire Western Africa 1963
## 5  Cote d'Ivoire Western Africa 1964
## 6  Cote d'Ivoire Western Africa 1965
## 7  Cote d'Ivoire Western Africa 1966
## 8  Cote d'Ivoire Western Africa 1967
## 9  Cote d'Ivoire Western Africa 1968
...
```

Les opérateurs logiques

opérateur	définition	opérateur	définition
<	plus petit	x | y	x OU y
<=	plus petit ou égale	is.na(x)	test si x est NA
>	plus grand que	!is.na(x)	test si x n'est pas NA
>=	plus grand ou égal à	x %in% y	test si x est contenue dans y
==	égale à	!(x %in% y)	test si x n'est pas contenue dans y
!=	différent de	!x	différent de x
x & y	x ET y		

Comment les opérateurs logiques fonctionnent




```
health %>%
  filter(country == "Cote d'Ivoire", year == 1960 | year == 2010)
```

```
##         country         region year
## 1 Cote d'Ivoire Western Africa 1960
## 2 Cote d'Ivoire Western Africa 2010
##   Births attended by skilled health staff (% of total)
...
```

Ordonner les lignes avec `arrange()`

`arrange()` permet d'ordonner les lignes de facon croissante ou décroissante.

```
health %>% arrange(year)
```

```
##                country          region year
## 1              Algeria Northern Africa 1545
## 2               Angola   Middle Africa 1545
## 3                Benin  Western Africa 1545
## 4             Botswana Southern Africa 1545
...
```

```
health %>% arrange(desc(year))
```

```
##                       country          region year
## 1                     Algeria Northern Africa 2099
## 2                      Angola   Middle Africa 2099
## 3                       Benin  Western Africa 2099
## 4                    Botswana Southern Africa 2099
...
```

Note: les valeurs NA sont toujours mises à la fin.

Sélectionner un certain nombre de ligne avec `slice()`

Pour les 10 premières lignes:

```
health %>% slice(1:10)
```

```
##    country          region year
## 1  Algeria Northern Africa 1545
## 2  Algeria Northern Africa 1586
## 3  Algeria Northern Africa 1590
## 4  Algeria Northern Africa 1610
## 5  Algeria Northern Africa 1614
## 6  Algeria Northern Africa 1635
## 7  Algeria Northern Africa 1644
## 8  Algeria Northern Africa 1657
## 9  Algeria Northern Africa 1660
...
```

Sélectionner des variables avec `select()`

- Sélectionner pour garder les variables

```
health %>% select(country, year, region)
```

```
##                       country year          region
## 1                     Algeria 1545 Northern Africa
## 2                     Algeria 1586 Northern Africa
## 3                     Algeria 1590 Northern Africa
## 4                     Algeria 1610 Northern Africa
...
```

- Exclure des variables

```
health %>% select(-region)
```

```
##           country year Births attended by skilled health staff (% of total)
## 1         Algeria 1545                                                    NA
## 2         Algeria 1586                                                    NA
## 3         Algeria 1590                                                    NA
## 4         Algeria 1610                                                    NA
...
```

Sélectionner des variables avec `select()`

Sélectionner une suite variables

```
health %>%
  select(country:region)
```

```
##                       country          region
## 1                     Algeria Northern Africa
## 2                     Algeria Northern Africa
## 3                     Algeria Northern Africa
## 4                     Algeria Northern Africa
## 5                     Algeria Northern Africa
## 6                     Algeria Northern Africa
## 7                     Algeria Northern Africa
## 8                     Algeria Northern Africa
## 9                     Algeria Northern Africa
...
```

Ajouter de nouvelles variables avec `mutate()`

`mutate()` permet d'ajouter de nouvelles colonnes (variables).

```
health %>%
  mutate(elapsed_years = 2020 - year)
```

```
##       country          region year
## 1     Algeria Northern Africa 1545
## 2     Algeria Northern Africa 1586
## 3     Algeria Northern Africa 1590
## 4     Algeria Northern Africa 1610
## 5     Algeria Northern Africa 1614
## 6     Algeria Northern Africa 1635
## 7     Algeria Northern Africa 1644
## 8     Algeria Northern Africa 1657
## 9     Algeria Northern Africa 1660
...
```

Mesures groupées avec `summarise()`

```
health %>%
  summarise(`Life expectancy` = mean(`Life expectancy`, na.rm = TRUE))
```

```
##   Life expectancy
## 1        37.40498
```

Combiné avec `group_by()`

```
health %>%
  group_by(region) %>%
  summarise(`Life expectancy` = mean(`Life expectancy`, na.rm = TRUE))
```

```
## # A tibble: 5 x 2
##   region          `Life expectancy`
##   <fct>                       <dbl>
## 1 Eastern Africa               37.7
## 2 Middle Africa                36.1
## 3 Northern Africa              40.5
## 4 Southern Africa              39.1
## 5 Western Africa               36.1
```

Autres fonctions utiles

- `pull()` pour extraire une variable comme vecteur

```
health %>% pull(region)
```

```
##     [1] Northern Africa Northern Africa Northern Africa Northern Africa
##     [5] Northern Africa Northern Africa Northern Africa Northern Africa
##     [9] Northern Africa Northern Africa Northern Africa Northern Africa
##    [13] Northern Africa Northern Africa Northern Africa Northern Africa
##    [17] Northern Africa Northern Africa Northern Africa Northern Africa
...
```

- `distinct()` pour avoir les lignes uniques

```
health %>% distinct(region)
```

```
##            region
## 1 Northern Africa
## 2   Middle Africa
## 3  Western Africa
## 4 Southern Africa
...
```

Autres fonctions utiles

- `sample_n()` pour un echantillon aléatoire de **n** éléments

```
health %>% sample_n(5, replace = FALSE)
```

```
##         country         region year
## 1       Liberia Western Africa 1855
## 2 Guinea-Bissau Western Africa 1872
## 3         Kenya Eastern Africa 1860
## 4        Rwanda Eastern Africa 1956
...
```

- `sample_frac()` pour un échantillon aléatoire de **frac** %

```
health %>% sample_frac(0.2, replace = FALSE)
```

```
##                     country          region year
## 1                   Eritrea  Eastern Africa 1815
## 2                    Gambia  Western Africa 1958
## 3                     Egypt Northern Africa 1710
## 4                    Malawi  Eastern Africa 2001
...
```

Autres fonctions utiles

- `count()` pour le nombre d'observation par groupe

```
health %>% count(region)
```

```
## # A tibble: 5 x 2
##   region             n
##   <fct>          <int>
## 1 Eastern Africa   4791
## 2 Middle Africa    2317
## 3 Northern Africa  1800
## 4 Southern Africa  1500
## 5 Western Africa   4800
```

Que faut-il retenir?

- C'est (très) facile et aisé de créer des graphiques beaux et complexes, et de manipuler des données dans R.
- La grammaire des graphiques utilise les couches pour créer les graphiques (le package ggplot2).
- Manipuler les données (avec tidyverse) reviens à se demander quelle action ont veut faire et utiliser le verbe correspondant.

Merci !