Manipuler ses données avec le tidyverse

Benjamin Louis

15/10/2019 (MàJ: 18/10/2019)


tidyverse
www.rstudio.com

Le tidyverse

- Ensemble de packages destinés à la science des données
- (Co)développés par Hadley Wickham, *chief scientist* chez 

- Philosophie, grammaire et structure de données communes :
  - *tidy data*
  - *layered grammar of graphics*
- Instalation complète :

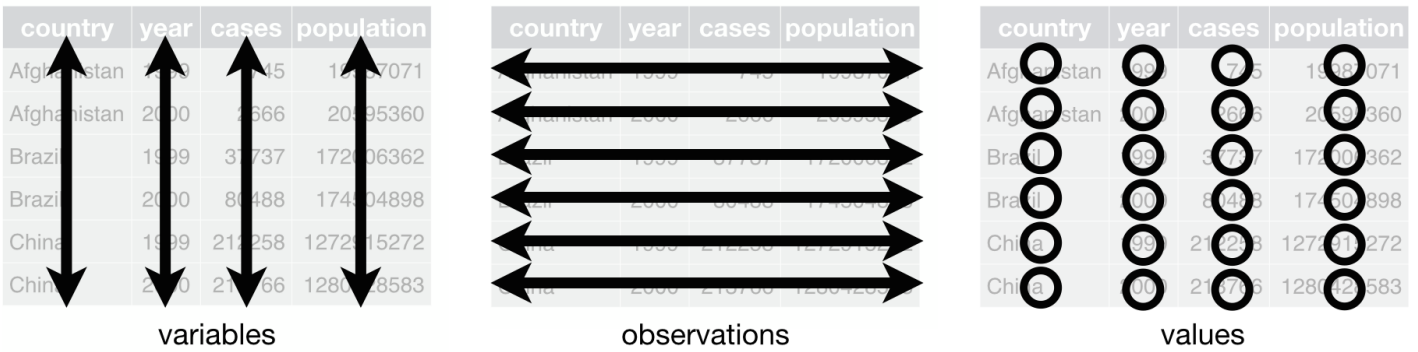
```
install.packages("tidyverse")
```

Le tidyverse

Le tidyverse

et plus encore ...

Le tidyverse : *tidy data*



Le tidyverse : *grammar*

- **Volonté d'obtenir un code compréhensible**
- **Les noms des fonction sont des verbes**
- **Syntaxe qui permet une lecture semblable à des phrases !**

%>% ou *pipe*

- Premier argument d'une fonction

```
# On peut remplacer :
verb(arg1 = objet, arg2 = "valeur_arg2")
# par :
objet %>%
  verb(arg2 = "valeur_arg2")
```

- Autre argument

```
# On peut remplacer :
verb(arg1 = valeur_arg_1, arg2 = objet)
# par :
objet %>%
  verb(arg1 = valeur_arg_1, arg2 = .)
```

%>% ou *pipe*

À la place de :

```
objet <- verb_1(objet, ...)
objet <- verb_2(objet, ...)
objet <- verb_3(objet, ...)
objet <- verb_4(objet, ...)
```

On peut écrire :

```
objet <- objet %>%
  verb_1(...) %>%
  verb_2(...) %>%
  verb_3(...) %>%
  verb_4(...)
```


TIBBLE
www.rstudio.com

tibble, un autre data.frame

> *On garde le bon, on jette le mauvais*

- Pas de changement du nom ou du type des variables
- une méthode `print()` améliorée

```
as_tibble(iris)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

tibble, création

Comme `data.frame()`

```
tibble(x = 1:5, y = 1,
       z = x ^ 2 + y)
```

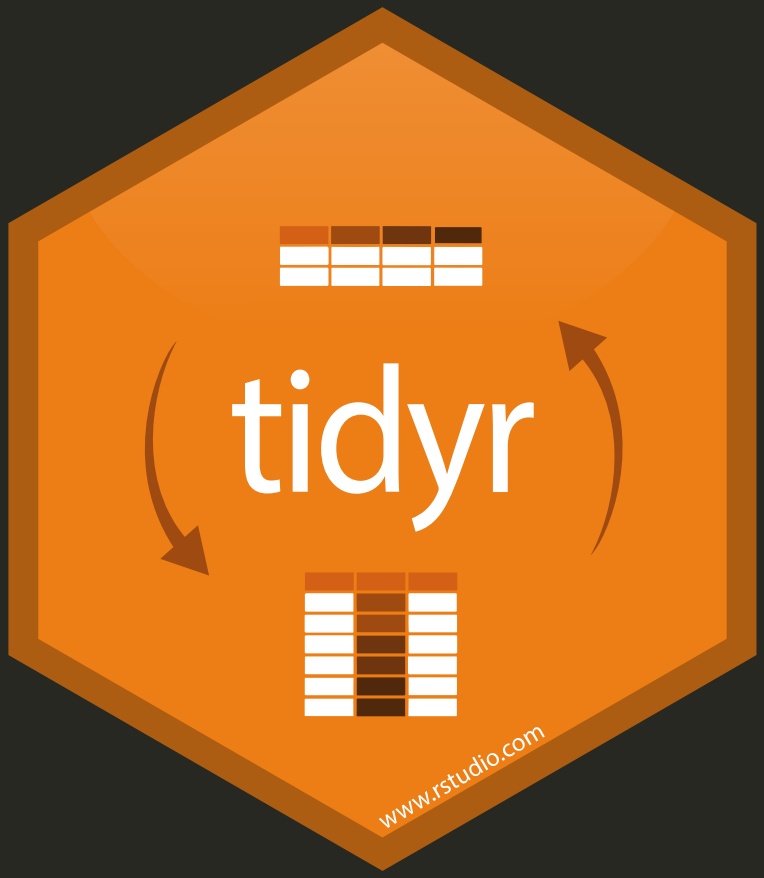
```
## # A tibble: 5 x 3
##       x     y     z
##   <int> <dbl> <dbl>
## 1     1     1     2
## 2     2     1     5
## 3     3     1    10
## 4     4     1    17
## 5     5     1    26
```

ou par ligne

```
tribble(
  ~x, ~y,  ~z,
  "a", 2,  3.6,
  "b", 1,  8.5
)
```

```
## # A tibble: 2 x 3
##   x         y     z
##   <chr> <dbl> <dbl>
## 1 a         2   3.6
## 2 b         1   8.5
```

*Les exemples proviennent de https://tibble.tidyverse.org/


tidyr
www.rstudio.com

tidyr::

- pivot_longer()
- pivot_wider()
- separate()
- unite()
- complete()
- drop_na()

tidyr::pivot_*

wide

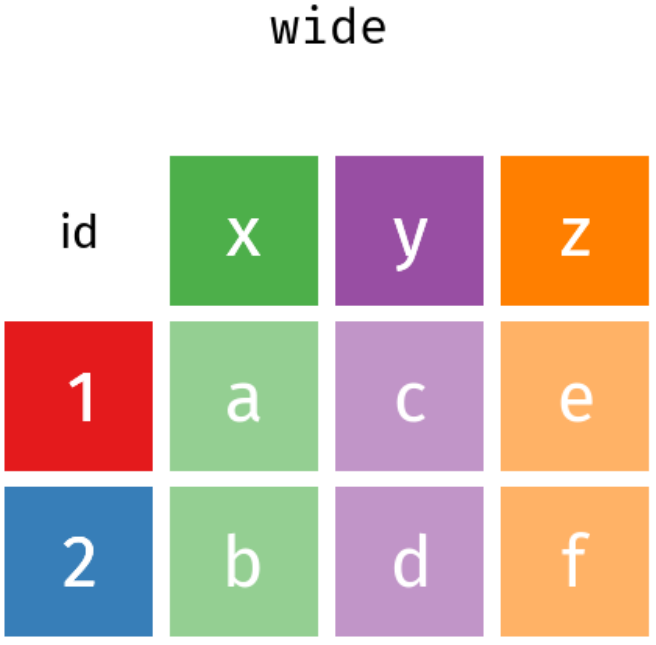
id	x	y	z
1	a	c	e
2	b	d	f

long

id	key	val
1	x	a
2	x	b
1	y	c
2	y	d
1	z	e
2	z	f

*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

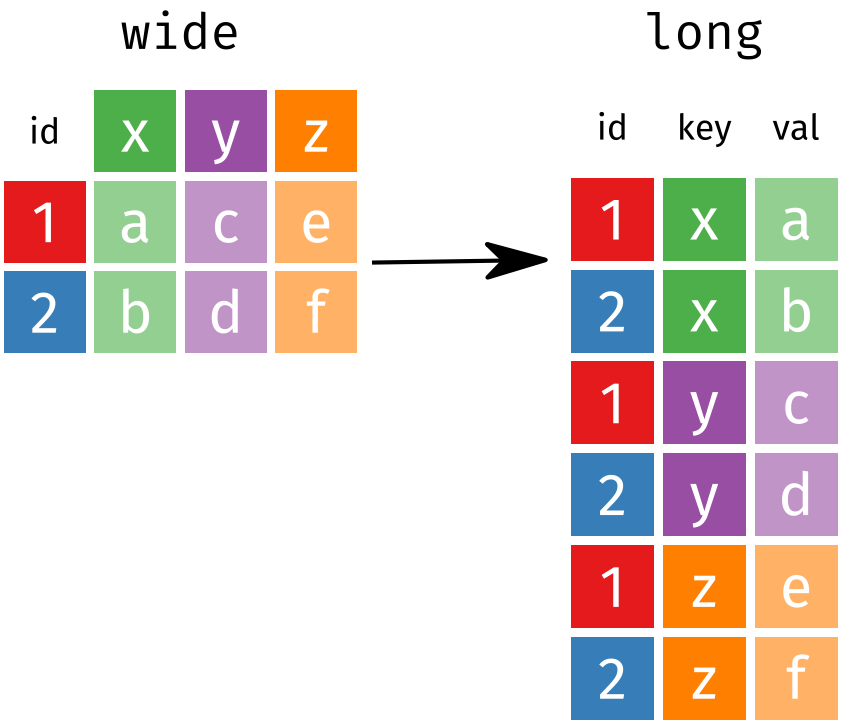
tidyr::pivot_*



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::pivot_longer()

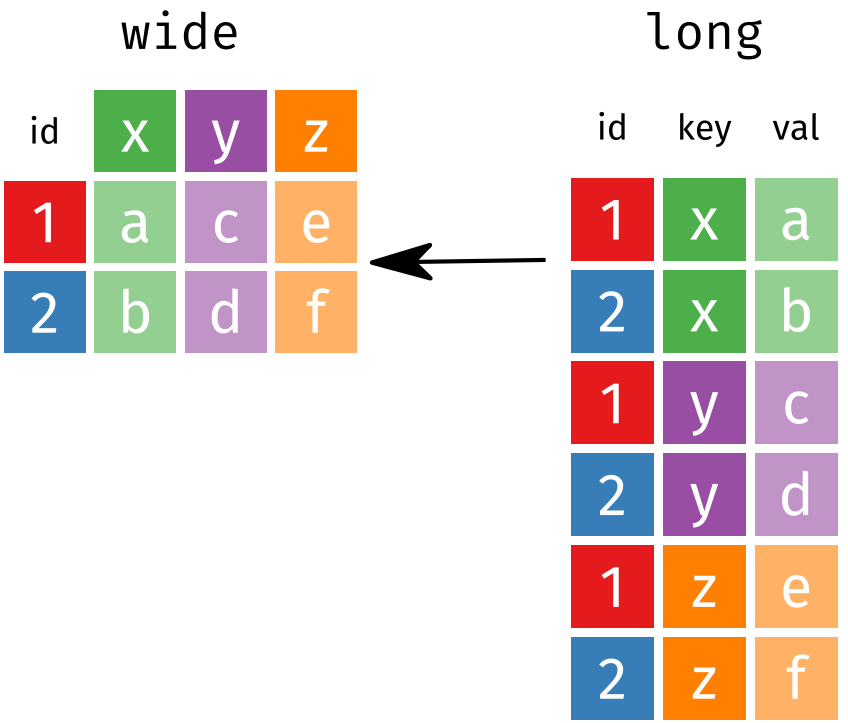
```
pivot_longer(wide, x:z, names_to = "key", values_to = "val")
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::pivot_wider()

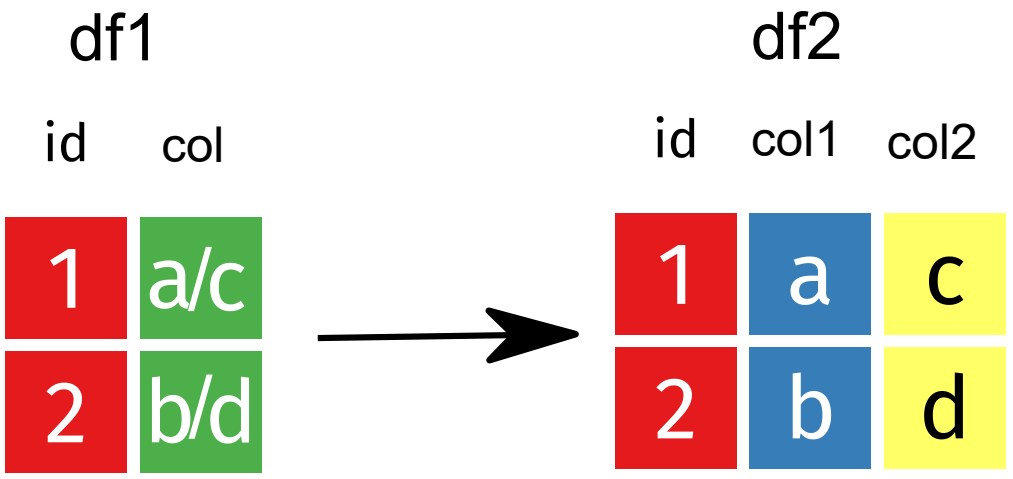
```
pivot_wider(long, id_cols = "id", names_from = "key", values_from = "val")
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::separate()

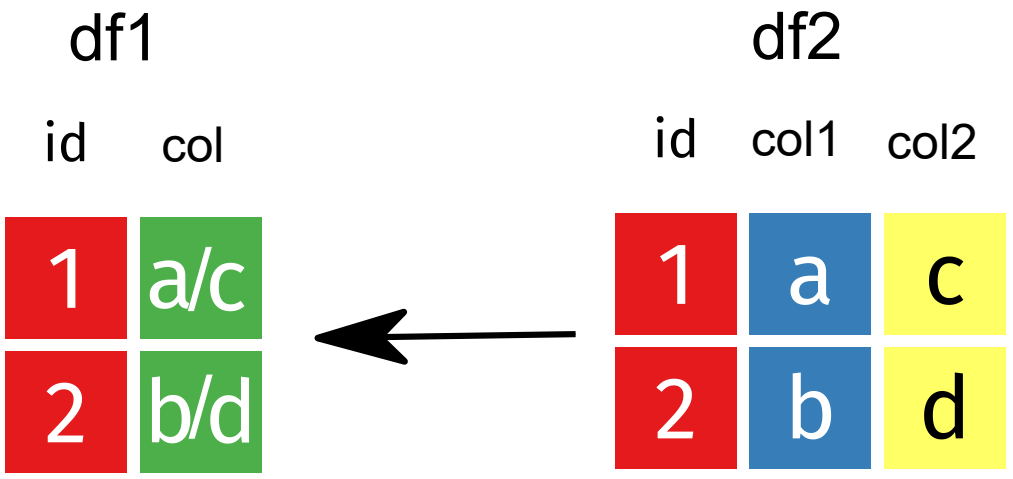
```
separate(df1, col = col, into = c("col1", "col2"), sep = "/")
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::unite()

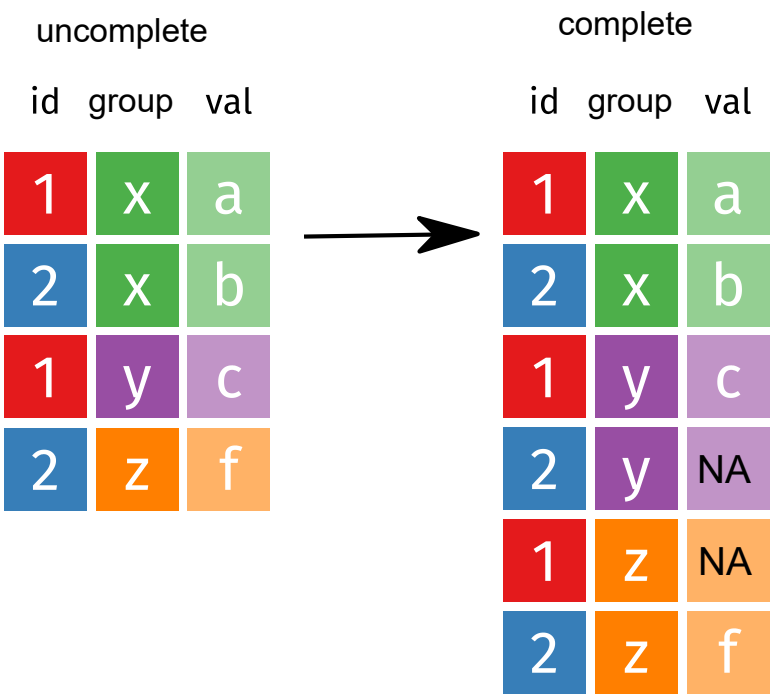
```
unite(df2, col = "col", col1, col2, sep = "/")
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::complete()

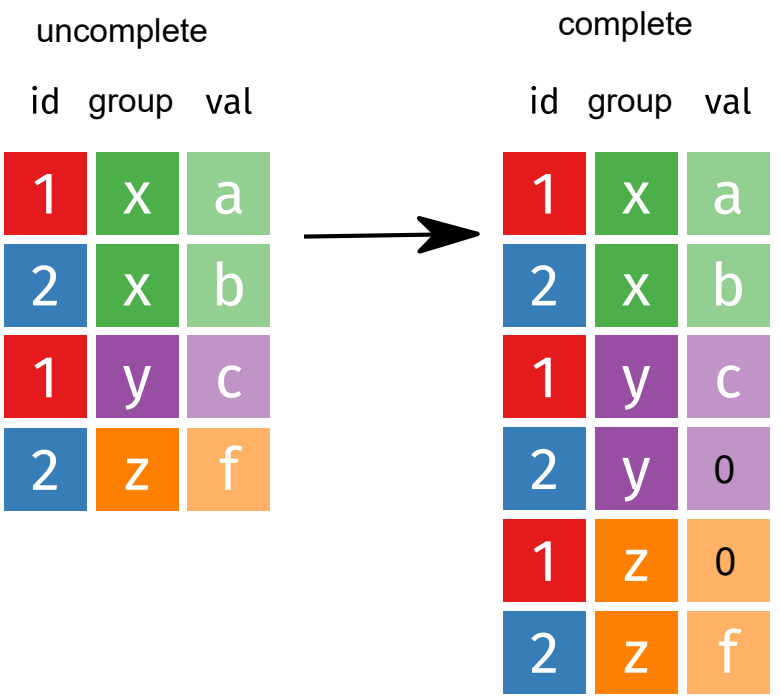
```
complete(uncomplete, id, group)
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::complete()

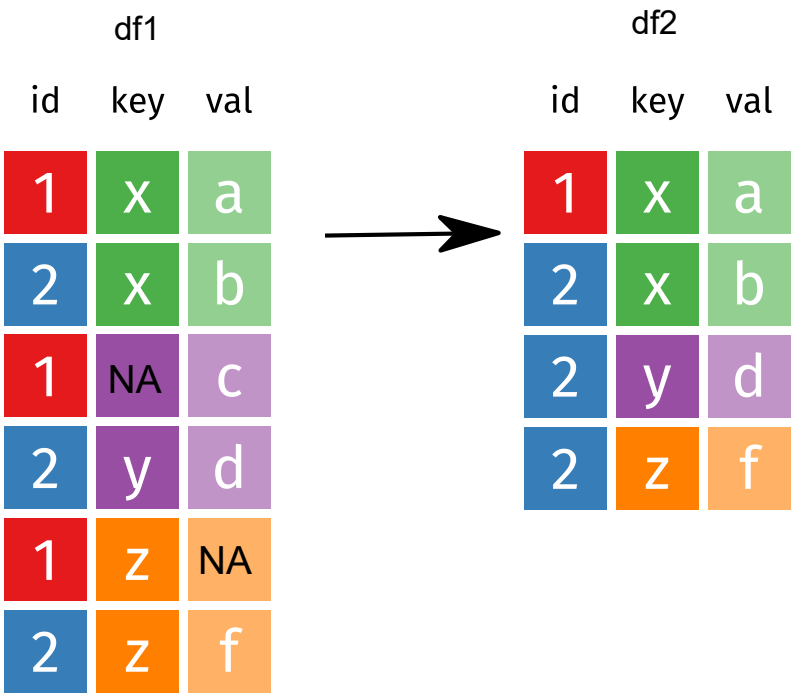
```
complete(uncomplete, id, group, fill = list(val = 0))
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::drop_na()

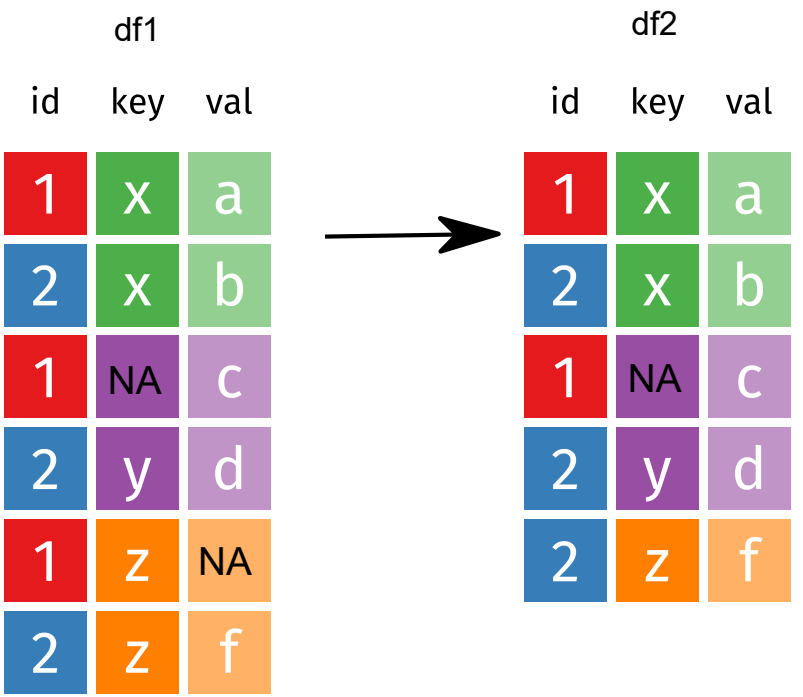
```
drop_na(df1)
```



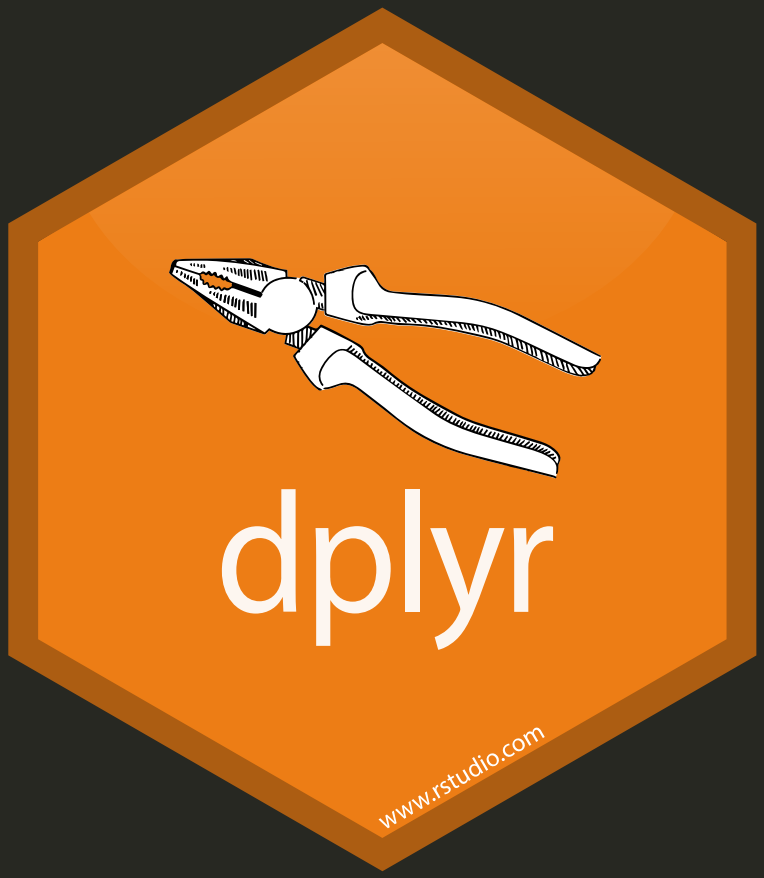
*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain

tidyr::drop_na()

```
drop_na(df1, val)
```



*Images et animations proviennent/adaptées de https://github.com/gadenbuie/tidyexplain


dplyr
www.rstudio.com

tidyr::

- `select()`
- `rename()`
- `mutate()`
- `filter()`
- `arrange()`
- `summarize()`
- `group_by()`

Jeu de données d'exemple : iris

```
iris <- as_tibble(iris)
iris
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

dplyr::select()

➔ Sélectionner des variables

```
select(iris, Sepal.Length, Species)
```

```
## # A tibble: 150 x 2
##    Sepal.Length Species
##           <dbl> <fct>
##  1          5.1 setosa
##  2          4.9 setosa
##  3          4.7 setosa
##  4          4.6 setosa
##  5          5   setosa
##  6          5.4 setosa
##  7          4.6 setosa
##  8          5   setosa
##  9          4.4 setosa
## 10          4.9 setosa
## # ... with 140 more rows
```

dplyr::select()

➔ Comment séléctionner*

```
# Bornes
select(iris, Sepal.Length:Species)

# on enlève Sepal.Length et Species
select(iris, -Sepal.Length, -Species)

# Commence par Sepal
select(iris, starts_with("Sepal"))

# Termine par Length
select(iris, starts_with("Length"))

# Contient Length
select(iris, contains("Length"))

# Species puis tout le reste
select(iris, Species, everything())

# On peut renommer
select(iris, sepal_length = Sepal.Length, Species)

# On peut tout combiner
select(iris, contains("Length"), -Sepal.Length)
```

*cf `?select`

dplyr::select_*()

select_all()

```
select_all(iris, .funs = tolower)
select_all(iris, .funs = list(~tolower(.))
```

select_at()

```
select_at(iris, .vars = vars(-Species))
select_at(iris, .vars = vars(-Species), .funs = tolower)
```

select_if()

```
select_if(iris, .predicate = is.numeric)
select_if(iris, .predicate = is.numeric, .funs = tolower)
```

`tolower()` : transforme les majuscules en minuscules

dplyr::rename()

➔ Renommer des variables

```
rename(iris, long_petal = Sepal.Length)
```

```
## # A tibble: 150 x 5
##    long_petal Sepal.Width Petal.Length Petal.Width Species
##         <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1        5.1         3.5          1.4         0.2 setosa
##  2        4.9         3            1.4         0.2 setosa
##  3        4.7         3.2          1.3         0.2 setosa
##  4        4.6         3.1          1.5         0.2 setosa
##  5        5           3.6          1.4         0.2 setosa
##  6        5.4         3.9          1.7         0.4 setosa
##  7        4.6         3.4          1.4         0.3 setosa
##  8        5           3.4          1.5         0.2 setosa
##  9        4.4         2.9          1.4         0.2 setosa
## 10        4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

dplyr::rename_*()

- rename_all()

```
rename_all(iris, .funs = tolower)
```

- rename_at()

```
rename_at(iris, .vars = vars(-Species), .funs = tolower)
```

- rename_if()

```
rename_if(iris, .predicate = is.numeric, .funs = tolower)
```

dplyr::mutate()

➔ Ajouter des variables

```
mutate(iris, taux_sepal = Sepal.Length/Sepal.Width)
```

```
## # A tibble: 150 x 6
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species taux_sepal
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>        <dbl>
##  1          5.1         3.5          1.4         0.2 setosa        1.46
##  2          4.9         3            1.4         0.2 setosa        1.63
##  3          4.7         3.2          1.3         0.2 setosa        1.47
##  4          4.6         3.1          1.5         0.2 setosa        1.48
##  5          5           3.6          1.4         0.2 setosa        1.39
##  6          5.4         3.9          1.7         0.4 setosa        1.38
##  7          4.6         3.4          1.4         0.3 setosa        1.35
##  8          5           3.4          1.5         0.2 setosa        1.47
##  9          4.4         2.9          1.4         0.2 setosa        1.52
## 10          4.9         3.1          1.5         0.1 setosa        1.58
## # ... with 140 more rows
```

Il existe plusieurs fonctions utiles : `?mutate`

dplyr::mutate_*()

- mutate_all()

```
mutate_all(iris, round)
```

- mutate_at()

```
mutate_at(iris, vars(contains("Sepal")), round)
```

- mutate_if()

```
mutate_if(iris, is.numeric,
          .funs = list(~ . - mean(., na.rm = TRUE)))
```

dplyr::filter()

➔ Filtrer les observations

```
filter(iris, Species == "versicolor" & Sepal.Length >= 6.5)
```

```
## # A tibble: 9 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          7           3.2          4.7         1.4 versicolor
## 2          6.9         3.1          4.9         1.5 versicolor
## 3          6.5         2.8          4.6         1.5 versicolor
## 4          6.6         2.9          4.6         1.3 versicolor
## 5          6.7         3.1          4.4         1.4 versicolor
## 6          6.6         3            4.4         1.4 versicolor
## 7          6.8         2.8          4.8         1.4 versicolor
## 8          6.7         3            5           1.7 versicolor
## 9          6.7         3.1          4.7         1.5 versicolor
```

dplyr::filter_*()

- filter_all()

```
filter_all(iris, .vars_predicate = all_vars(!is.na(.)))
```

- filter_at()

```
filter_at(iris, .vars = vars(starts_with("Se")),
          .vars_predicate = any_vars(. > mean(.)))
```

- filter_if()

```
filter_if(iris, .predicate = is.numeric,
          .vars_predicate = all_vars(. > mean(.)))
```

dplyr::arrange()

➔ Classer les observations selon des variables

```
arrange(iris, desc(Species), Sepal.Length)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.9         2.5          4.5         1.7 virginica
##  2          5.6         2.8          4.9         2   virginica
##  3          5.7         2.5          5           2   virginica
##  4          5.8         2.7          5.1         1.9 virginica
##  5          5.8         2.8          5.1         2.4 virginica
##  6          5.8         2.7          5.1         1.9 virginica
##  7          5.9         3            5.1         1.8 virginica
##  8          6           2.2          5           1.5 virginica
##  9          6           3            4.8         1.8 virginica
## 10          6.1         3            4.9         1.8 virginica
## # ... with 140 more rows
```

dplyr::arrange_*()

arrange_all()

```
arrange_all(iris)
arrange_all(iris. .funs = )
```

arrange_at()

```
arrange_at(iris, .vars = vars(contains("Sepal")))
arrange_at(iris, .vars= vars(contains("Sepal")), .funs = desc)
```

arrange_if()

```
arrange_if(iris, .predicate = ~!is.numeric(.))
arrange_if(iris, .predicate = ~!is.numeric(.), .funs = desc)
```

dplyr::summarize()

➔ Synthétiser des variables

```
summarize(iris, moy = mean(Sepal.Length))
```

```
## # A tibble: 1 x 1
##     moy
##   <dbl>
## 1  5.84
```

Intéressant surtout avec `group_by()` !!!

dplyr::summarize_*()

- summarize_all()

```
summarize_all(iris, n())
```

- summarize_at()

```
summarize_at(iris, .vars = vars(-Species), .funs = mean)
```

- summarize_if()

```
summarize_if(iris, .predicate = is.numeric, .funs = mean)
```

dplyr::group_by()

➲ Grouper en fonction de variables (`dplyr::ungroup()` pour dégrouper)

```
group_by(iris, Species)
```

```
## # A tibble: 150 x 5
## # Groups:   Species [3]
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

Peut se combiner avec beaucoup d'autres fonctions !

dplyr::group_by() + filter()

➔ Grouper en fonction de variables

```
group_by(iris, Species) %>%
  filter_if(is.numeric, ~. > mean(.))
```

```
## # A tibble: 30 x 5
## # Groups:   Species [3]
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.4         3.9          1.7         0.4 setosa
##  2          5.7         4.4          1.5         0.4 setosa
##  3          5.7         3.8          1.7         0.3 setosa
##  4          5.1         3.8          1.5         0.3 setosa
##  5          5.1         3.7          1.5         0.4 setosa
##  6          5.1         3.8          1.9         0.4 setosa
##  7          7           3.2          4.7         1.4 versicolor
##  8          6.4         3.2          4.5         1.5 versicolor
##  9          6.9         3.1          4.9         1.5 versicolor
## 10          6.5         2.8          4.6         1.5 versicolor
## # ... with 20 more rows
```

dplyr::group_by() + summarize()

➔ Grouper en fonction de variables

```
group_by(iris, Species) %>%
  summarize_if(is.numeric, .funs = mean)
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

dplyr::group_by() + mutate()

➔ Grouper en fonction de variables

```
group_by(iris, Species) %>%
  mutate(moy_sepal_length = mean(Sepal.Length)) %>%
  select(Species, moy_sepal_length)
```

```
## # A tibble: 150 x 2
## # Groups:   Species [3]
##    Species moy_sepal_length
##    <fct>              <dbl>
##  1 setosa              5.01
##  2 setosa              5.01
##  3 setosa              5.01
##  4 setosa              5.01
##  5 setosa              5.01
##  6 setosa              5.01
##  7 setosa              5.01
##  8 setosa              5.01
##  9 setosa              5.01
## 10 setosa              5.01
## # ... with 140 more rows
```

Exercices

Dans un script :

```
library(tidyr)
library(dplyr)
who <- who %>%
  rename_at(vars(starts_with("new")),
            ~stringr::str_replace(., "new_?", ""))
```

1. Transformer **who** en *tidy data*
2. Ajouter une variable avec le nombre total de cas dans chaque pays chaque année
3. Quelle année a connu le plus de cas dans le mondeClasser dans l'ordre croissant par ce nombre total
4. Comparer le nombre de cas par tranche d'âge en France

Correction

```
tidywho <- who %>%
  # 1. Tidy
  pivot_longer(sp_m014:rel_f65, names_to = "key", values_to = "cas") %>%
  drop_na(cas) %>%
  separate(key, into = c("type", "sexage"), sep = "_") %>%
  separate(sexage, into = c("sex", "age"), sep = 1) %>%
  mutate(age = recode(age, "014" = "0-14", "1524" = "15-24",
                      "2534" = "25-34", "3544" = "35-44",
                      "4554" = "45-54", "5564" = "55-64", "65" = "65+")) %>%
  select(-iso2, -iso3) %>%
  # 2. Total par pays et annee
  group_by(country, year) %>%
  mutate(total = sum(cas)) %>%
  ungroup()
```

```
## # A tibble: 76,046 x 7
##    country      year type  sex   age     cas total
##    <chr>       <int> <chr> <chr> <chr> <int> <int>
##  1 Afghanistan  1997 sp    m     0-14      0   128
##  2 Afghanistan  1997 sp    m     15-24    10   128
##  3 Afghanistan  1997 sp    m     25-34     6   128
##  4 Afghanistan  1997 sp    m     35-44     3   128
##  5 Afghanistan  1997 sp    m     45-54     5   128
##  6 Afghanistan  1997 sp    m     55-64     2   128
##  7 Afghanistan  1997 sp    m     65+       0   128
##  8 Afghanistan  1997 sp    f     0-14      5   128
##  9 Afghanistan  1997 sp    f     15-24    38   128
## 10 Afghanistan  1997 sp    f     25-34    36   128
## # ... with 76,036 more rows
```

Correction

```
# 3. Année avec le plus de cas
tidywho %>%
  group_by(year) %>%
  summarise(total = sum(cas)) %>%
  arrange(desc(total))
```

```
## # A tibble: 34 x 2
##     year   total
##    <int>   <int>
##  1  2010 3986811
##  2  2011 3977151
##  3  2012 3962589
##  4  2007 3836674
##  5  2009 3834194
##  6  2008 3535278
##  7  2013 3220572
##  8  2006 2996524
##  9  2005 2364023
## 10  2004 2184924
## # ... with 24 more rows
```

Correction

```
# 4. Comparaison tranche d'âge en France
tidywho %>%
  filter(country == "France") %>%
  group_by(age) %>%
  summarize(pmoy = sum(total*cas)/sum(total))
```

```
## # A tibble: 7 x 2
##   age    pmoy
##   <chr> <dbl>
## 1 0-14   36.9
## 2 15-24  99.2
## 3 25-34 172.
## 4 35-44 144.
## 5 45-54 117.
## 6 55-64  90.7
## 7 65+   199.
```

Références

- https://www.tidyverse.org/
- https://thinkr.fr/tidyverse-hadleyverse/
- https://r4ds.had.co.nz/
- http://vita.had.co.nz/papers/tidy-data.html
- http://vita.had.co.nz/papers/layered-grammar.html
- https://github.com/gadenbuie/tidyexplain