

A photograph of a modern building with a glass facade and a courtyard. The building has a white brick wall on the left and a glass facade on the right. The courtyard is filled with green plants and has a paved path. The sky is blue with a few clouds.

Manipulation de données avec dplyr

Vincent Guillemot
Amoury Vaysse
Mardi 16/11/2021

INSTITUT Pasteur

OMICS

Enchaîner les commandes avec magrittr

- On utilise un opérateur

$\%>\%$

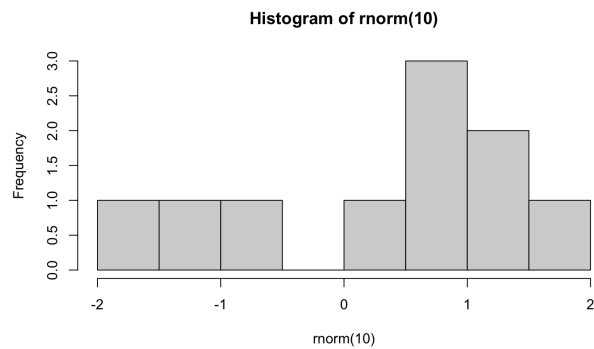


- Avant : $f(g(x))$
- Après : $g(x) \%>\% f()$

Exemple

```
set.seed(7895)
```

```
hist(rnorm(10))
```



```
set.seed(7895)
```

```
library(magrittr)
```

```
#>
```

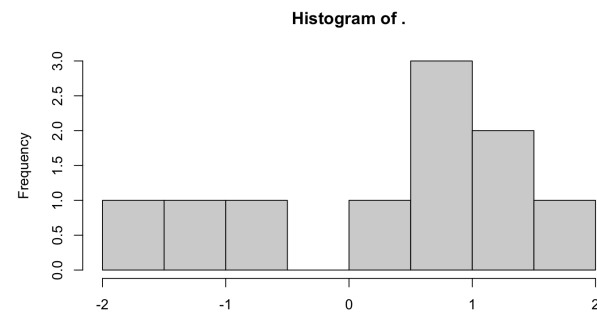
```
#> Attachement du package : 'magrittr'
```

```
#> L'objet suivant est masqué depuis 'package:t
```

```
#>
```

```
#>      extract
```

```
rnorm(10) %>% hist()
```



Le tidyverse



Tidyverse: <https://www.tidyverse.org/>

Avant toute chose

Charger le package dplyr...

```
library(dplyr) # ou require(dplyr)
```

Ou bien charger tidyverse...

```
library(tidyverse)
```

... mais cela chargera d'autres packages en plus

Et charger les données.

```
data("fruits", package = "debuter")
```

Le format “tibble”

Les données sont au format “tibble” : c’est comme des “data-frames” mais en mieux !

```
fruits
#> # A tibble: 51 × 18
#>   nom      groupe Energie   Eau Proteines Glucides Lipides Sucres
#>   <chr>    <chr>    <dbl> <dbl>    <dbl>    <dbl>  <dbl>  <dbl>
#> 1 Abricot  crus      194  87.1     0.81     9.01   0.25   6.7
#> 2 Abricot  secs     1010  24.7     2.88    59.1   0.5    34.3
#> 3 Ananas   exoti...   304  81.3     0.94    15.1   0.25   14.9
#> 4 Banane   exoti...   383  75.8     1.06    19.7   0.25   15.6
#> 5 Canneberge secs     1410  14.6     0.25    76.4   1      72.8
#> 6 Cerise   crus      235  85.7     0.81    13     0.25   10
#> 7 Citron   crus      118  91.3     0.25    1.56   0.25   0.8
#> 8 Clementine crus      200  87      0.81    9.17   0.25   8.6
#> 9 CompoteMul... compo...   279  82.9     0.25    15.3   0.08   14.6
#> 10 CompotePom... compo...   432  72.9     0.23    24.4   0.21   20.7
#> # ... with 41 more rows, and 10 more variables: Fructose <dbl>,
#> #   Fibres <dbl>, Calcium <dbl>, Magnesium <dbl>, Phosphore <dbl>,
#> #   Potassium <dbl>, Zinc <dbl>, BetaCarotene <dbl>,
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Les fonctions de dplyr

Nous allons voir ensemble quelques fonctions très pratiques de la librairie `dplyr`.

#	Fonction (US)	Fonction (UK)	Description
1	<code>mutate</code>	<code>mutate</code>	Créer ou modifier des colonnes
2	<code>select</code>	<code>select</code>	Sélectionner des colonnes
3	<code>arrange</code>	<code>arrange</code>	Trier les lignes
4	<code>filter</code>	<code>filter</code>	Sélectionner des lignes
5	<code>group_by</code>	<code>group_by</code>	Grouper des lignes
6	<code>summarize</code>	<code>summarise</code>	Résumer des groupes
7	<code>count</code>	<code>count</code>	Compter

Créer ou modifier des colonnes

Avec la fonction `mutate`.

```
fruits2 <- fruits %>%  
  mutate(Sucres_ratio = Sucres / 100)  
  
head(fruits2[, "Sucres_ratio"])  
#> # A tibble: 6 × 1  
#>   Sucres_ratio  
#>       <dbl>  
#> 1      0.067  
#> 2      0.343  
#> 3      0.149  
#> 4      0.156  
#> 5      0.728  
#> 6      0.1
```

Avec les fonctions classiques.

```
fruits2 <- fruits  
fruits2$Sucres_ratio <-  
  fruits2$Sucres / 100  
  
head(fruits2[, "Sucres_ratio"])  
#> # A tibble: 6 × 1  
#>   Sucres_ratio  
#>       <dbl>  
#> 1      0.067  
#> 2      0.343  
#> 3      0.149  
#> 4      0.156  
#> 5      0.728  
#> 6      0.1
```

Sélectionner des colonnes

Avec la fonction `select`.

```
fruits %>%  
  select(  
    Energie,  
    Sucres,  
    Lipides,  
    Proteines)
```

Avec les fonctions classiques.

```
fruits[,  
  c(  
    "Energie",  
    "Sucres",  
    "Lipides",  
    "Proteines") ]
```

Sélectionner des colonnes - bis

la fonction `select` est très versatile !

```
fruits %>%  
  select(Energie:Proteines, - Sucres)
```

On peut sélectionner des plages entières de colonnes sur la base de leurs noms, en enlever avec le `-`, combiner tout cela avec la fonction `c()` ... ou pas !

Attention, la flexibilité a un coût !

Trier des lignes

Avec les fonctions `arrange` et `desc`.

```
fruits %>%
  select(Energie, Sucres, Fibres) %>%
  arrange(desc(Fibres))
#> # A tibble: 51 × 3
#>   Energie Sucres Fibres
#>   <dbl>   <dbl> <dbl>
#> 1    1010    34.3    8.3
#> 2     425     8.5    6.8
#> 3    1410    72.8    5.7
#> 4     198     6.1    5.2
#> 5     969    38.1    5.1
#> 6     289     6.63   4.6
#> 7     206     5.4    4.3
#> 8     170     2.1    4.3
#> 9    1360    70.3    4.2
#> 10     293    12.2    4.1
#> # ... with 41 more rows
```

Avec les fonctions classiques

```
fruits[
  order(fruits$Fibres, decreasing = TRUE),
  c("Energie", "Sucres", "Fibres")]
#> # A tibble: 51 × 3
#>   Energie Sucres Fibres
#>   <dbl>   <dbl> <dbl>
#> 1    1010    34.3    8.3
#> 2     425     8.5    6.8
#> 3    1410    72.8    5.7
#> 4     198     6.1    5.2
#> 5     969    38.1    5.1
#> 6     289     6.63   4.6
#> 7     206     5.4    4.3
#> 8     170     2.1    4.3
#> 9    1360    70.3    4.2
#> 10     293    12.2    4.1
#> # ... with 41 more rows
```

Sélectionner des lignes

Avec la fonction `filter`.

```
fruits %>%
  filter(Sucres > 60)
#> # A tibble: 2 × 18
#>   nom          groupe Energie   Eau Proteines
#>   <chr>        <chr>    <dbl> <dbl>    <dbl>
#> 1 Canneberge secs      1410  14.6    0.25
#> 2 Raisin      secs      1360  16      3
#> # ... with 10 more variables: Fructose <dbl>,
#> #   Calcium <dbl>, Magnesium <dbl>, Phosphor
#> #   Potassium <dbl>, Zinc <dbl>, BetaCaroter
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Avec les fonctions classiques.

```
fruits[fruits$Sucres > 60, ]
#> # A tibble: 2 × 18
#>   nom          groupe Energie   Eau Proteines Glucides L
#>   <chr>        <chr>    <dbl> <dbl>    <dbl>    <dbl>
#> 1 Canneberge secs      1410  14.6    0.25    76.4
#> 2 Raisin      secs      1360  16      3      73.2
#> # ... with 10 more variables: Fructose <dbl>, Fibres <db
#> #   Calcium <dbl>, Magnesium <dbl>, Phosphore <dbl>,
#> #   Potassium <dbl>, Zinc <dbl>, BetaCarotene <dbl>,
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Sélectionner des plages de lignes

Avec la fonction `slice`.

```
fruits %>%
  slice(3:10)
#> # A tibble: 8 × 18
#>   nom          groupe Energie   Eau Proteine
#>   <chr>        <chr>   <dbl> <dbl>   <dbl>
#> 1 Ananas      exoti...   304  81.3    0.9
#> 2 Banane      exoti...   383  75.8    1.0
#> 3 Canneberge secs      1410  14.6    0.2
#> 4 Cerise      crus       235  85.7    0.8
#> 5 Citron      crus       118  91.3    0.2
#> 6 Clementine crus       200  87      0.8
#> 7 CompoteMult... compo...   279  82.9    0.2
#> 8 CompotePomme compo...   432  72.9    0.2
#> # ... with 10 more variables: Fructose <dbl>,
#> #   Calcium <dbl>, Magnesium <dbl>, Phosphor
#> #   Potassium <dbl>, Zinc <dbl>, BetaCaroter
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Avec les fonctions classiques.

```
fruits[3:10, ]
#> # A tibble: 8 × 18
#>   nom          groupe Energie   Eau Proteines Glucides
#>   <chr>        <chr>   <dbl> <dbl>   <dbl>   <dbl>
#> 1 Ananas      exoti...   304  81.3    0.94    15.1
#> 2 Banane      exoti...   383  75.8    1.06    19.7
#> 3 Canneberge secs      1410  14.6    0.25    76.4
#> 4 Cerise      crus       235  85.7    0.81     13
#> 5 Citron      crus       118  91.3    0.25    1.56
#> 6 Clementine crus       200  87      0.81    9.17
#> 7 CompoteMult... compo...   279  82.9    0.25    15.3
#> 8 CompotePomme compo...   432  72.9    0.23    24.4
#> # ... with 10 more variables: Fructose <dbl>, Fibres <db
#> #   Calcium <dbl>, Magnesium <dbl>, Phosphore <dbl>,
#> #   Potassium <dbl>, Zinc <dbl>, BetaCarotene <dbl>,
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Grouper des lignes

Avec la fonction `group_by` :

```
fruits %>% group_by(groupe)
#> # A tibble: 51 × 18
#> # Groups:   groupe [4]
#>   nom      groupe Energie   Eau Proteines Glucides Lipides Sucres
#>   <chr>    <chr>    <dbl> <dbl>    <dbl>    <dbl>    <dbl>  <dbl>
#> 1 Abricot   crus      194  87.1     0.81     9.01     0.25    6.7
#> 2 Abricot   secs     1010  24.7     2.88    59.1     0.5     34.3
#> 3 Ananas    exoti...   304  81.3     0.94    15.1     0.25    14.9
#> 4 Banane    exoti...   383  75.8     1.06    19.7     0.25    15.6
#> 5 Canneberge secs     1410  14.6     0.25    76.4     1       72.8
#> 6 Cerise    crus      235  85.7     0.81     13      0.25    10
#> 7 Citron    crus      118  91.3     0.25     1.56    0.25     0.8
#> 8 Clementine crus      200  87      0.81     9.17    0.25     8.6
#> 9 CompoteMul... compo...   279  82.9     0.25    15.3     0.08    14.6
#> 10 CompotePom... compo...   432  72.9     0.23    24.4     0.21    20.7
#> # ... with 41 more rows, and 10 more variables: Fructose <dbl>,
#> #   Fibres <dbl>, Calcium <dbl>, Magnesium <dbl>, Phosphore <dbl>,
#> #   Potassium <dbl>, Zinc <dbl>, BetaCarotene <dbl>,
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Les données sont prêtes à être “traitées” groupe par groupe. PS : L’opération `ungroup()` permet d’enlever les groupes.

Calculer une moyenne

Avec la fonction `summarize`.

```
fruits %>%
  group_by(groupe) %>%
  summarize(SucreMoyen = mean(Sucres))
#> # A tibble: 4 × 2
#>   groupe   SucreMoyen
#>   <chr>     <dbl>
#> 1 compote    15.5
#> 2 crus       9.68
#> 3 exotique   11.4
#> 4 secs      55.0
```

Avec les fonctions classiques.

```
aggregate(fruits$Sucres,
          by = list(fruits$groupe),
          FUN = mean)
#>   Group.1      x
#> 1  compote 15.533333
#> 2   crus  9.684242
#> 3 exotique 11.380000
#> 4   secs 54.980000
```


Exercice(s)

Calculer l'énergie moyenne, la teneur en sucres médiane et le maximum de la teneur en Fibres par groupe de fruits et trier le tout par ordre décroissant du maximum de la teneur en Fibres !

Deux autres fonctions pour sélectionner ou transformer des colonnes

	Sélectionne	Ne sélectionne pas
Ne transforme pas	<code>select</code>	<code>rename</code>
Peut transformer	<code>transmute</code>	<code>mutate</code>



David Robinson

Principal Data Scientist at Heap

Compter

Avec le “verbe” count :

```
fruits %>% count(groupe)
#> # A tibble: 4 × 2
#>   groupe      n
#>   <chr>   <int>
#> 1 compote     3
#> 2 crus       33
#> 3 exotique    10
#> 4 secs        5
```

On peut ensuite ranger les résultats par ordre décroissant :

```
fruits %>%
  count(groupe) %>%
  arrange(desc(n))
#> # A tibble: 4 × 2
#>   groupe      n
#>   <chr>   <int>
#> 1 crus       33
#> 2 exotique    10
#> 3 secs        5
#> 4 compote     3
```

Compter deux choses à la fois

Par exemple, compter dans chaque groupe le nombre de fruits dont la teneur en Vitamine C est inférieure ou supérieure à 50 :

```
fruits %>%  
  mutate(VitCqual = cut(VitamineC, c(0, 50, 100))) %>%  
  count(groupe, VitCqual, name = "N")  
#> # A tibble: 6 × 3  
#>   groupe    VitCqual      N  
#>   <chr>    <fct>    <int>  
#> 1 compote (0,50]      3  
#> 2 crus    (0,50]     32  
#> 3 crus    (50,100]    1  
#> 4 exotique (0,50]      7  
#> 5 exotique (50,100]    3  
#> 6 secs    (0,50]      5
```

Super bonus : la table de contingence

Ce n'est pas facile, il vaut mieux utiliser la fonction `table` :

```
library(tidyr)
fruits %>%
  mutate(VitCqual = cut(VitamineC, c(0, 50, 100))) %>%
  count(groupe, VitCqual, name = "N") %>%
  pivot_wider(id_cols = groupe,
              names_from = VitCqual,
              values_from = N)

#> # A tibble: 4 × 3
#>   groupe   `(0,50]` `(50,100]`
#>   <chr>     <int>     <int>
#> 1 compote         3         NA
#> 2 crus           32          1
#> 3 exotique        7          3
#> 4 secs           5         NA
```

Avec les illustrations de Allison Horst
(<https://www.allisonhorst.com/>)

D'autres fonctions utiles en bonus

dplyr::^{1.0.0}relocate()
move COLUMNS around!

Default: move to FRONT
or move to
.before or .after
A SPECIFIED COLUMN!



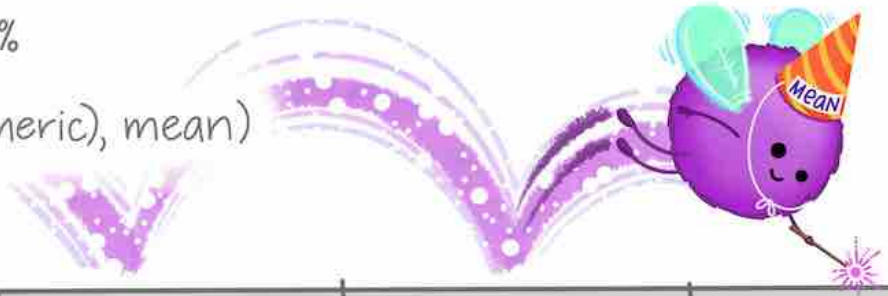
@allison_horst

dplyr::across()

use within `mutate()`
or `summarize()` to
apply function(s) to
a selection of columns!

EXAMPLE:

```
df %>%  
  group_by(species) %>%  
  summarize(  
    across(where(is.numeric), mean)  
  )
```



species	mass_g	age_yr	range_sqmi
pika	163	2.4	0.46
marmot	1509	3.0	0.87
marmot	2417	5.6	0.62

@allison_horst

dplyr::case_when()

IF ELSE...
(but you love it?)

df %>% ^{ADD COLUMN 'danger'}

mutate(danger = case_when(
 IF type is kraken THEN danger is extreme!
 type == "kraken" ~ "extreme!",
 TRUE ~ "high"))
OTHERWISE, danger is high.



Data Wrangling with dplyr and tidyr

Cheat Sheet



Il y a tellement d'autres fonctions !
