

# Tidyverse : Manipulation des données

Eric Marcon

14 février 2024

# Manifeste

# Approche complète de l'analyse de données

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Données bien rangées (*tidy*).

Enchaînement des opérations (`|>` de *magrittr*, + de *ggplot2*).

Programmation fonctionnelle (pas orientée objet), optimisée pour les utilisateurs (lisibilité plutôt que performance).

```
library("tidyverse")  
vignette("manifesto", package="tidyverse")
```

Ensemble de packages, appelés par *tidyverse*

# Données rectangulaires

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Modèle du dataframe : une ligne par observation, une colonne par attribut.

Dataframe optimisé : tibble.

Documentation : vignette("tibble",  
package="tibble").

```
ggplot2::diamonds
```

```
## # A tibble: 53,940 x 10
##   carat cut      color clarity depth table price
##   <dbl> <ord>    <ord>  <ord>    <dbl> <dbl> <int>
## 1  0.23 Ideal    E      SI2      61.5    55   326
## 2  0.21 Premium E      SI1      59.8    61   326
## 3  0.23 Good     E      VS1      56.9    65   327
## 4  0.29 Premium I      VS2      62.4    58   334
## 5  0.31 Good     J      SI2      63.3    58   335
## # i 53,935 more rows
## # i 3 more variables: x <dbl>, y <dbl>, z <dbl>
```

# Pipe (tuyau)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Le package *magrittr* introduit le pipe `%>%` (Ctrl+Shift + m).



Modèle du pipeline de la programmation système repris par la bioinformatique.

## Exemple :

```
1:10 %>% sum()
```

```
## [1] 55
```

Principe : les données résultant d'un calcul sont passées à la fonction suivante.

## Enchaînement :

```
1:10 %>% sqrt() %>% sum()
```

```
## [1] 22.46828
```

Code plus lisible que `sum(sqrt(1:10))`.

Tuyau avec retour :

```
library("magrittr")
x <- c(4,9)
x %<>% sqrt()
x
```

```
## [1] 2 3
```

Embranchement :

```
x %T>% plot %>% sum
```



```
## [1] 5
```

Exposition :

```
diamonds %$% mean(price)
```

```
## [1] 3932.8
```

*# Équivalent à*

```
diamonds %>% pull(price) %>% mean()
```

```
## [1] 3932.8
```

Le tuyau de base est fourni aussi par *dplyr*, chargé par *tidyverse*, contrairement à *magrittr*.

Les autres sont peu utilisés, à éviter.



L'opérateur tuyau a été intégré à R à partir de la version 4.1.

Sa syntaxe est `|>`. Il est pratiquement identique à `%>%` mais ne nécessite pas de charger *dplyr* ou *magrittr*.

Dans R Studio, choisir dans “Tools > Global Options > Code” : cocher ou non “Use native pipe operator”.

## Bagarre (*Wrangling*) :

- Importation des données ;
- Nettoyage (*Tidy*) ;
- Transformation.

## Visualisation.

Communication : RMarkdown et sorties graphiques. Lire :

- Graphics for communication
- Top 50 ggplot2 Visualizations

# Bagarre

Lecture de fichiers texte variés.

Importation dans un tibble.

Référence

Fonctions `read_csv()` et `read_csv2()`.

Remplacent `read.csv()` et `read.csv2()` de base.

Plus rapide que les fonctions originales.

country	year	cases	population
Afghanistan	1999	18215	15467071
Afghanistan	2000	18166	20065360
Brazil	1999	31737	172036362
Brazil	2000	81488	174034898
China	1999	21358	1272015272
China	2000	21366	128003583

variables

country	year	cases	population
Afghanistan	1999	18215	15467071
Afghanistan	2000	18166	20065360
Brazil	1999	31737	172036362
Brazil	2000	81488	174034898
China	1999	21358	1272015272
China	2000	21366	128003583

observations

country	year	cases	population
Afghanistan	1999	18215	15467071
Afghanistan	2000	18166	20065360
Brazil	1999	31737	172036362
Brazil	2000	81488	174034898
China	1999	21358	1272015272
China	2000	21366	128003583

values

Approche habituelle en écologie (analyse multivariée par exemple).

Si les données sont mal rangées (“pas tidy”), quelques manipulations de base.

Référence

Données : inventaire d'une parcelle de Paracou, 4 carrés distincts.

Lire les données :

```
paracou6 <- read_csv2("data/Paracou6.csv")
```

- Afficher paracou6

# Rassemblement (*unite*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Famille, genre et espèce des arbres sont dans 3 colonnes.

Créer une colonne avec le nom complet de l'espèce.

```
library("dplyr")
paracou6 |>
  unite(col = spName, Family, Genus, Species, remove = FALSE) ->
  paracou6
```

- Afficher le résultat.

Le pipeline |> (Ctrl + Shift + m) passe la donnée à la fonction suivante. L'affectation finale -> enregistre le résultat.

La commande classique est :

```
paracou6 <- unite(data = paracou6, col = spName,
                  Family, Genus, Species, remove = FALSE)
```



# Séparation (*separate*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

## Opération contraire

country	year	rate
Afghanistan	1999	<b>745</b> / 19987071
Afghanistan	2000	<b>2666</b> / 20595360
Brazil	1999	<b>37737</b> / 172006362
Brazil	2000	<b>80488</b> / 174504898
China	1999	<b>212258</b> / 1272915272
China	2000	<b>213766</b> / 1280428583

table3

country	year	cases	population
Afghanistan	1999	<b>745</b>	19987071
Afghanistan	2000	<b>2666</b>	20595360
Brazil	1999	<b>37737</b>	172006362
Brazil	2000	<b>80488</b>	174504898
China	1999	<b>212258</b>	1272915272
China	2000	<b>213766</b>	1280428583



# Rassembler des colonnes (*pivot\_longer*)

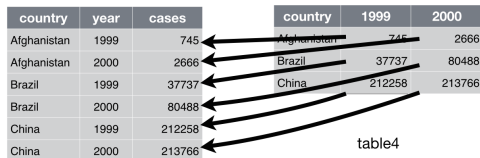
Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

## Opération inverse de la création d'un tableau croisé



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

# Séparer des colonnes (*pivot\_wider*)

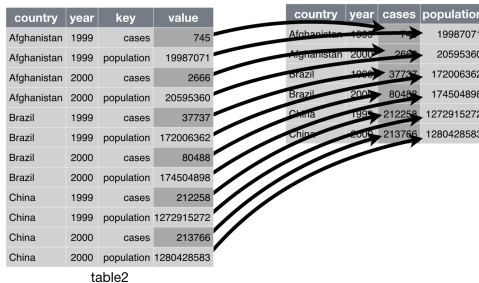
Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Crée une colonne par modalité d'une variable



country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Les valeurs manquantes explicites (valeur NA) peuvent être conservées dans les manipulations ou simplement supprimées avec l'option `na.rm=TRUE`.

`complete(var1, var2)` ajoute des enregistrements pour toutes les combinaisons de `var1` et `var2` manquantes.

Référence

## Outils du package *dplyr*

Idée :

- enchaîner les opérations de transformation avec les `|>` ;
- les écrire et les tester une à une.

# Filtrer les lignes (*filter*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

## Filtrer par des conditions sur les différentes variables

```
# Nombre de lignes
```

```
paracou6 |> count() |> pull()
```

```
## [1] 3541
```

```
# Après filtrage
```

```
paracou6 |> filter(SubPlot == 1) |> count() |> pull()
```

```
## [1] 942
```

Remarquer : `pull()` qui extrait la valeur finale du tibble de taille 1x1 produit par `count()`.

# Sélectionner les colonnes (*select*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Ne retenir que les colonnes intéressantes

```
paracou6 |>  
  select(SubPlot:Yfield, Family:Species, CircCorr) |>  
  ncol()
```

```
## [1] 8
```

Remarquer : `ncol()` est une fonction de *base*, pas du tidyverse.

# Ajouter des variables calculées (*mutate*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Des colonnes sont ajoutées au tibble

```
paracou6 |>
  select(idTree, CircCorr) |>
  mutate(Diametre = CircCorr/pi) |>
  print() ->
  paracou6_diam
```

```
## # A tibble: 3,541 x 3
##   idTree CircCorr Diametre
##   <dbl>    <dbl>    <dbl>
## 1 100655      44      14.0
## 2 100657     43.5     13.8
## 3 100658     53.5     17.0
## 4 100659     38.5     12.3
## 5 100660      77     24.5
## # i 3,536 more rows
```



# Trier les lignes (*arrange*)

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Afficher les plus gros arbres de la parcelle :

```
paracou6_diam |> arrange(desc(CircCorr))
```

```
## # A tibble: 3,541 x 3
##   idTree CircCorr Diametre
##   <dbl>    <dbl>    <dbl>
## 1 104455     318     101.
## 2 103939     317     101.
## 3 102249     300.     95.3
## 4 102086     290     92.3
## 5 100904     288.     91.8
## # i 3,536 more rows
```

# Regrouper et résumer

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre

Quel est le diamètre moyen des arbres par famille ?

```
paracou6 |>
  group_by(Family) |>
  summarise(diam_mean = mean(CircCorr) / pi, trees_n = n()) |>
  arrange(desc(diam_mean))
```

```
## # A tibble: 51 x 3
##   Family          diam_mean trees_n
##   <chr>          <dbl>    <int>
## 1 Vochysiaceae    49.2         11
## 2 Combretaceae    48.4          2
## 3 Phyllanthaceae  41.9          3
## 4 Humiriaceae     38.6         20
## 5 Goupiaceae      37.4         19
## # i 46 more rows
```

`bind_cols()` et `bind_rows()`.

```
t1 <- tibble(col2 = c("A", "B"), col3 = 3:4)
tibble(col1 = 1:2) |> bind_cols(t1)
```

```
## # A tibble: 2 x 3
##   col1 col2   col3
##   <int> <chr> <int>
## 1     1   A       3
## 2     2   B       4
```

Équivalent de `cbind()` et `rbind()`

`inner_join()`, `left_join()`, `right_join()` et  
`full_join()`

```
tibble(col2 = c("B", "C"), col5 = 5:6) |> inner_join(t1)
```

```
## # A tibble: 1 x 3
##   col2    col5  col3
##   <chr> <int> <int>
## 1 B         5      4
```

Tidyverse :  
Manipulation  
des données

Eric Marcon

Manifeste

Bagarre