Organiser et traiter plus facilement ses données avec R : utiliser le package dplyr

Elodie BARIL - Arnaud BRINGE

INED – Service Méthodes Statistiques

7 février 2019

Séminaire RUSS

Plan

- Package dplyr, historique et contexte
- Les objets R traités, retour sur les data frames et leurs constituants

- Opérations élémentaires permises par dplyr
- Chaînage des opérations élémentaires. Exemples
- Calculs de statistiques agrégées
- Jointures et concaténations

- Transformation de data frames (opérations gather et spread)
- Opérations sur les métadonnées

Rappels et introduction à dplyr

Rappels de programmation R

- Fonctionnement par package
- Types de données
- Notion de data frame
- Accès aux éléments d'un data frame

Package dplyr

- Package développé par Hadley Wickham,(R Studio), qui a aussi développé ggplot2
- Utilisation d'une grammaire {dplyr}, composée essentiellement de 5 verbes qui vont indiquer des opérations courantes sur les lignes et les colonnes d'une structure de données
- Ne s'utilise que sur des dataframes (ou tibbles)
- Partie du tidyverse

Opérations élémentaires

Opérations élémentaires

- Un parallèle avec Sas (ou Stata) :
  - →Sélection d'observations (*if, where …*)
  - →Sélection de variables (*keep*)
  - →Création de variable (*étape data*)
  - →Transformation de table (*transpose, reshape*)
  - →Fusion de tables (*merge, joins sql*)

Opérations élémentaires en R

- Programmation R classique :
  - →Sélection d'observations (*subset*)
  - →Sélection de variables (*select, …*)
  - →Création de variable
  - →Transformation de table (*reshape*)
  - →Fusion de tables (*sql, merge*)

Syntaxes SAS et R classique comparées

```
data a ;
set in.hdv2003 ;
keep id age sexe poids ;

if age<40 then age_cl=1 ;
else age_cl=0 ;

if age>20 then output ;
run;
```

```
a=hdv2003[hdv2003$age>20,c("id",
"age","sexe","poids")]

a$age_cl=ifelse(a$age<40,"1","0")
```

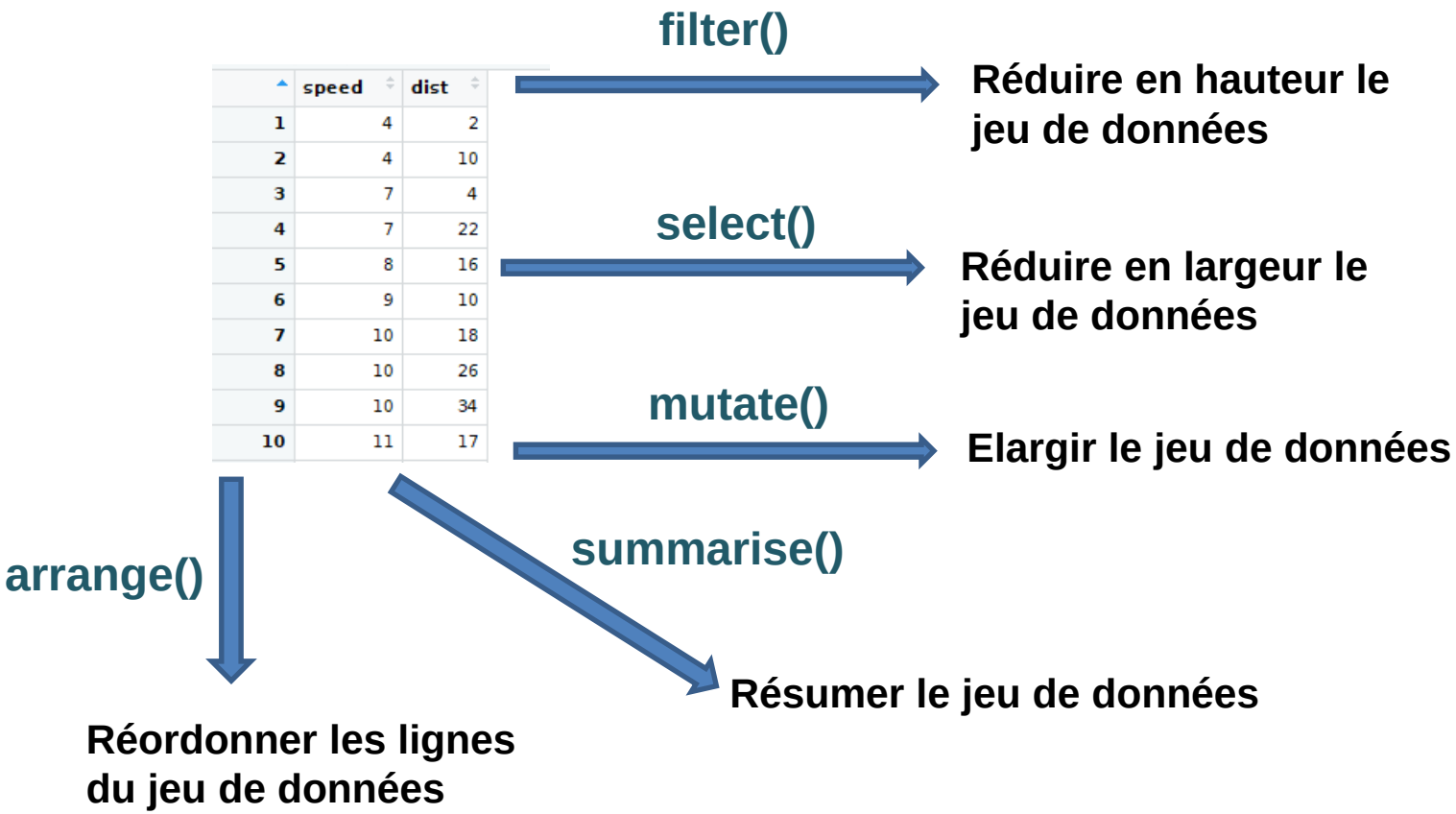
```
a = subset(hdv2003,age>20,
select=(id,age,sexe,poids))
```

```
a$age_cl=0
a[which(a$age<40),"age_cl"]=1
```

Fonctions de base : 5 verbes

- select(): Sélection de colonnes (variables)
- filter(): Sélection de lignes (observations)
- arrange(): Réordonnancement des lignes du dataframe
- mutate(): Ajout de colonnes (création de variables)
- summarise(): Production de statistiques résumées

Actions induites par dplyr



	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
9	10	34
10	11	17

Eléments de structure

- Arguments :
  1. Sur quel data frame je travaille ?
  2. Que dois je faire avec ?
  3. Résultat sous la forme d'un data frame resultant

- Exemple : `a = filter(hdv2003,age>20)`

- Opérateurs permis : ==, <, >, !=, %in%
- Opérateurs logiques : &, |, !
- Test sur valeur manquante : is.na()

Fonction de base : select()

select() permet de définir une liste de variables à sélectionner dans le data frame résultant

Syntaxe : select(*dataframe,liste_de_variables*)

- Liste de variables séparées par des “,” ou :
- Facilités d’écriture :
  - starts_with()
  - ends_with
  - contains()
  - matches()
  - one_of()
  - everything()

Fonction de base : select()

Exemples :

```
a=select(hdv2003,id,age,sexe)
a=select(hdv2003,id:clso)
a=select(hdv2003,id,starts_with("trav"))
a=select(hdv2003,id,age,contains("."))

exclus=c("age","sexe")
a=select(hdv2003,-one_of(exclus))

a=select(hdv2003,matches("^[a-z]{4}\\."))
```

Fonction de base : filter()

filter() permet de définir une liste d'observations sélectionnées en regard de conditions

Exemples :

```
a=filter(hdv2003,age>30)

a= filter(hdv2003, age >= 20 & age <= 30)

a=filter(hdv2003, qualif %in% c("Ouvrier specialise",
"Ouvrier qualifie"), age >= 20 & age <= 30)
```

Fonction de base : arrange()

arrange() permet de réordonner les observations en fonction de clés de tris.

Remarques :

- Une option desc permet de trier par valeurs décroissantes
- Il est possible d'indiquer plusieurs clés d'ordonnancement

Exemples :

```
a=arrange(hdv2003,sexe,age)
a=arrange(hdv2003,desc(age))
```

Fonction de base : mutate()

mutate() permet de créer une variable en fonction de variables existantes.

Attention aux NA

Exemples :

```
# Exemples mutate
a=mutate(hdv2003,age_cl=ifelse(age>40,"1","2"))
a=mutate(hdv2003,cadre=(qualif=="Cadre"))
```

```
a=mutate(hdv2003,csp_rg=ifelse(as.numeric(qualif) %in%
c(1,2), "Ouvrier", "Autre"))
```

Fonction de base : mutate()

case_when() permet de créer une variable en fonction de variables existantes, construction à plusieurs modalités

Exemples :

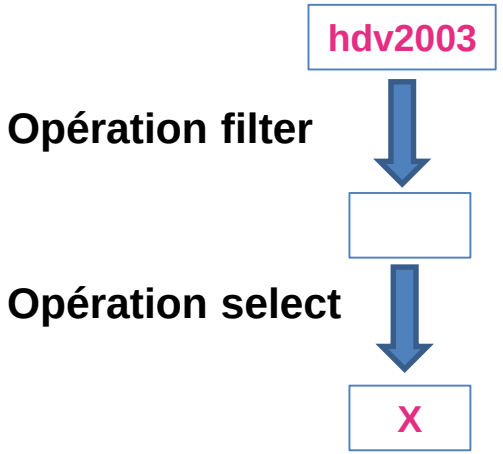
```
a=mutate(hdv2003,csp_rg=case_when(
  as.numeric(qualif) %in% c(1,2)   ~ "Ouvrier",
  as.numeric(qualif) %in% c(3,4,6) ~ "Technicien,
Interm, Employé",
  as.numeric(qualif)==5 ~ "Cadre",
  as.numeric(qualif)==7 ~ "Autre"))
```

Package dplyr : %>%

- Chaînage des opérations élémentaires (« then »)
- Fonctionnalité du package dplyr, issu du package magrittr

```
X = select(filter(hdv2003, sexe == "Femme"),
id, age, poids, occup)
```

```
X = hdv2003 %>%
filter(sexe == "Femme") %>%
select(id, age, poids, occup)
```



Syntaxes SAS et R dplyr

```
data a ;
set in.hdv2003 ;
keep id age sexe poids ;

if age<40 then age_cl=1 ;
else age_cl=0 ;

if age>20 then output ;
run;
```

```
a = hdv2003 %>%
filter(age>20) %>%
select(id,age,sexe,poids) %>%
mutate(age_cl=ifelse(age>40,"1","2"))
```

Syntaxes R base et R dplyr

```
a <- hdv2003[hdv2003$age>20, c("id", "age", "sexe",  "poids")]
a$age_cl <- ifelse(a$age>40,"1","2")
```

```
a = hdv2003 %>%
filter(age>20) %>%
select(id,age,sexe,poids) %>%
mutate(age_cl=ifelse(age>40,"1","2"))
```

Fonction de base : group_by

La fonction group_by s'utilise avec les %>% et permet de définir des groupes de lignes à partir des valeurs d'une ou plusieurs colonnes

Exemples :

```
# creation de variable avec group by:
hdv2= hdv2003 %>%
  group_by(sexe) %>%
  mutate(mean_age = mean(age, na.rm = TRUE)) %>%
  select(id,sexe,age,mean_age)

# group_by peut aussi être utile avec filter :
test=hdv2003 %>%
  group_by(sexe) %>%
  filter(age == max(age, na.rm = TRUE))
```

Fonction de base : summarise()

summarise() permet de produire des statistiques à un niveau agrégé. Il faut indiquer le(s) critère(s) d'agrégation et la (les) statistique(s) à calculer.

Syntaxe :

- Une instruction group_by est indispensable pour specifier en amont les critères d'aggregation
- Quelques exemples de function : count, mean, sum, max
- Il est possible de préciser des conditions dans les calculs (exemple sum(x>10)
- Le résultat peut être stocké au niveau individuel (mutate) ou agrégé (summarise)
- Une option na.rm=T permet d'éliminer les observations pour lesquelles figure une valeur manquante.
- Une instruction ungroup() peut être necessaire pour réinitaliser le calcul de variables individuelles.

Fonction de base : summarise()

Exemples :

```
a = hdv2003 %>%
  group_by(sexe) %>%
  summarise(moy_age=mean(age))
```

Le résultat peut aussi être stocké à un niveau individuel

```
a = hdv2003 %>%
  group_by(sexe) %>%
  mutate(moy_age=mean(age))
```

Fonction de base : summarise()

On peut introduire des conditions dans les calculs effectués par summarise

```
a = hdv2003 %>%
  group_by(sexe) %>%
  summarise(moy_age=mean(age[as.numeric(qualif)=="Cadre"],
  na.rm=T))
```

équivalent à

```
X = hdv2003 %>%
  filter(as.numeric(qualif)==5) %>%
  group_by(sexe) %>%
  summarise(m=mean(age,na.rm=T))
```

Fonction de base : summarise()

Une variable de rang peut être calculée, selon une variable de coupure

```
a = hdv2003 %>%
  arrange(age,id) %>%
  group_by(age) %>%
  mutate(rang=row_number())
```

Produire un tableau de fréquence avec dplyr

- **Tableaux de fréquences (dplyr)**

```
res = hdv2003 %>%
  group_by(sexe, qualif) %>%
  summarise (n = n()) %>%
  group_by(sexe) %>%
  mutate(som=sum(n)) %>%
  mutate(freq = n / som)


# (ALTERNATIVE)
res = hdv2003 %>%
  group_by(sexe, qualif) %>%
  summarise (n = n()) %>%
  ungroup() %>%
  mutate(freq = n / sum(n))
```

Autres opérations et fonctions

- distinct() : Sélection d'une seule observation par clé, uniquement les variables mentionnées. L'option .keep_all=T permet de conserver l'ensemble des variables.

- lag() et lead() : Création d'une variable à partir d'une information issue de l'observation précédente (vs suivante)

Les jointures et concaténations

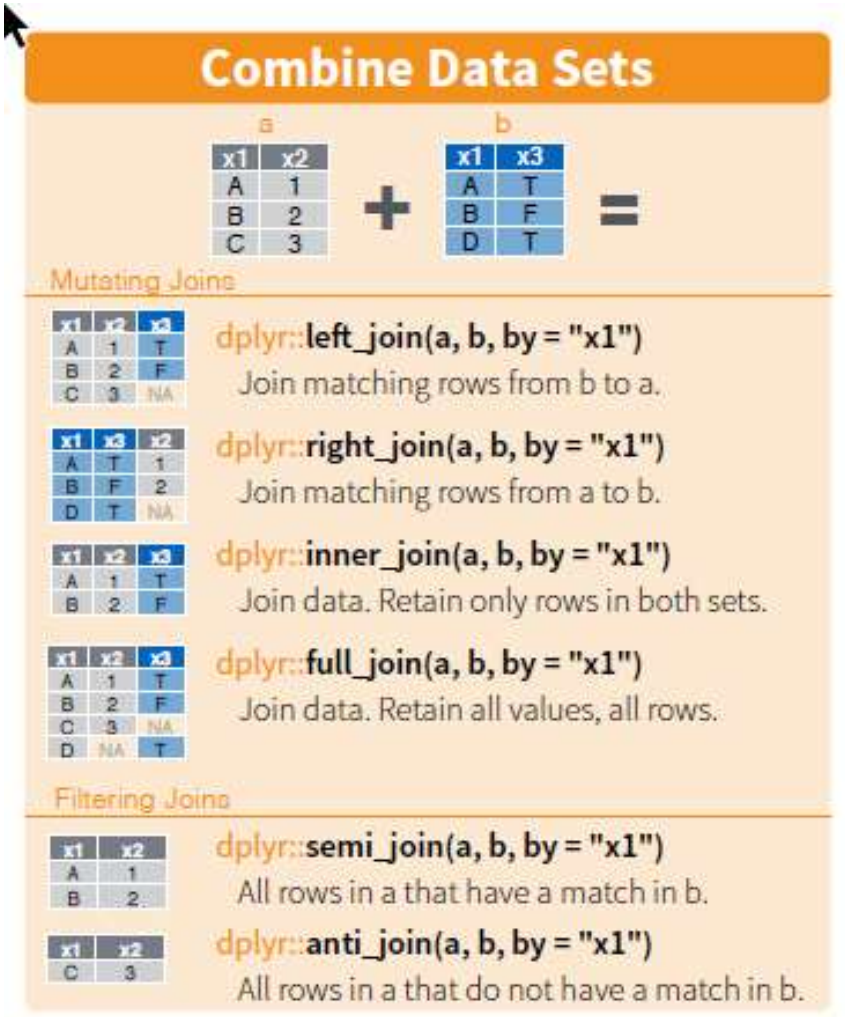
Les jointures

- Syntaxe générale
  - → Type_de_jointure(dataframes, by="cle")

- Eléments de programmation
  - → Dataframes
  - → Clés de jointure. (Peuvent être de noms différents)

Les différents types de jointures

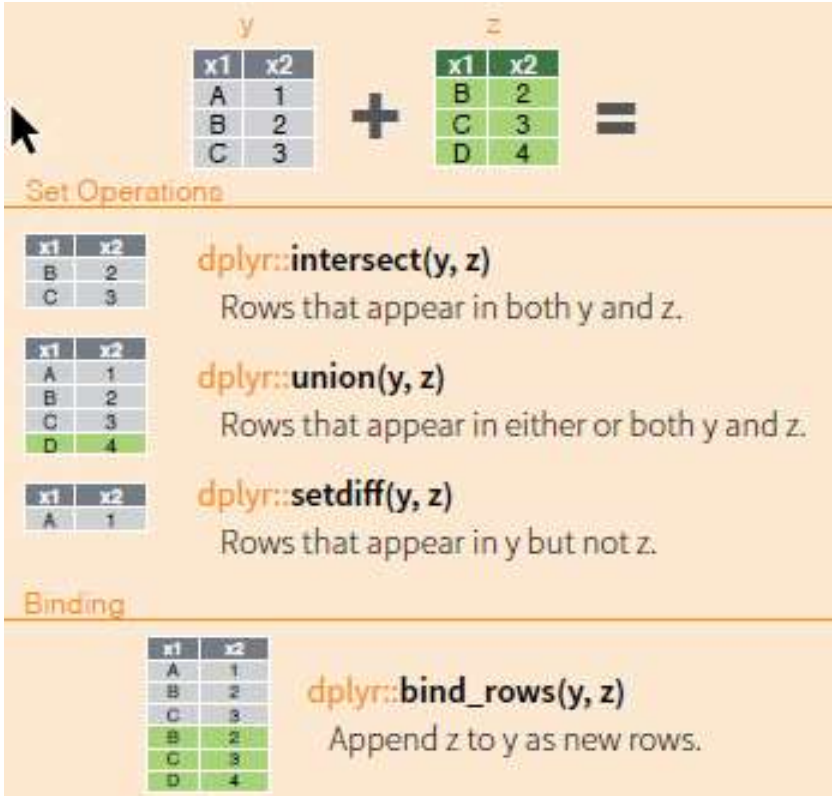
- Inner join
- Full join
- Left (right) join
- Semi join
- Anti join



Source : Rstudio cheat sheet

Joindre des observations

- bind_rows()
- intersect()
- setdiff()
- union()



Source : Rstudio cheat sheet

Les jointures et empilements

- Limites des jointures
  - → Variables de même nom conservées dans le dataframe de sortie (avec suffixe .x et .y par défaut)
  - → Pas de possibilité de lier directement 3 dataframes via dplyr

- Limites des empilements
  - → Variables identiques uniquement (sauf bind_rows)

Transformation de bases

Introduction

- Tidy datas : Par définition, une donnée est représentée par une valeur sur une ligne (observation) pour une mesure (variable). Sinon, il faut transformer les données.

- Exemple :

country	1992	1997	2002	2007
Belgium	10045622	10199787	10311970	10392226
France	57374179	58623428	59925035	61083916
Germany	80597764	82011073	82350671	82400996

country	annee	population
Belgium	1992	10045622
France	1992	57374179
Germany	1992	80597764
Belgium	1997	10199787
France	1997	58623428
Germany	1997	82011073
Belgium	2002	10311970
France	2002	59925035
Germany	2002	82350671
Belgium	2007	10392226
France	2007	61083916
Germany	2007	82400996

Verbes tidys datas, wide to long (gather)

- Permet de restructurer un dataframe par information (1 ligne = 1 information, et non un individu)

gather(data, key, value, ..., na.rm = FALSE, convert = FALSE, factor_key = FALSE**)**

gather() moves column names into a **key** column, gathering the column values into a single **value** column.

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

Source : Rstudio cheat sheet

Reshaping data : From wide to long

- **Exemple :** Comment représenter graphiquement une évolution de population pour plusieurs pays ?

Tableau A.1. Population au 1er janvier (milliers)

	1980	1990	2000	2007	2008	2009	2010
Europe du Nord							
Danemark	5 120	5 135	5 330	5 447	5 476	5 511	5 535
Finlande	4 771	4 974	5 171	5 277	5 300	5 326	5 351
Islande	227	254	279	308	315	319	318
Norvège	4 079	4 233	4 478	4 681	4 737	4 799	4 855
Suède	8 303	8 527	8 861	9 113	9 183	9 256	9 341

Source : Avdeev, A., Eremenko, T., Festy, P., Gaymu, J., Le Bouteillec, N., & Springer, S. (2011). Populations et tendances démographiques des pays européens (1980-2010). *Population*, *66*(1), 9-133.

Verbes tidys datas, long to wide (spread)

- Permet de restructurer un dataframe par individu (1 ligne = 1 individu, et non une information)

spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

spread() moves the unique values of a **key** column into the column names, spreading the values of a **value** column across the new columns.

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

key value

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

spread(table2, type, count)

Source : Rstudio cheat sheet

Reshaping data :

- From long to wide
  - → Exemple : Comment passer d'une observation niveau logement à une observation niveau individu ?

- From wide to long
  - Exemple : Comment passer d'une observation niveau individu (vecteur de positions résidentielles) à une observation niveau logement ?

Traitements massifs

Application à un groupe de variables (1/2)

- Syntaxe : summarise_at, summarise_all

L'objectif est de pouvoir appliquer plusieurs fonctions à plusieurs variables.

- Exemple :

  → Quelles sont les populations minimales, moyennes et maximales des populations 1968 à 1990, par département ?

Application à un groupe de variables (2/2)

- Syntaxe : mutate_at, mutate_if

- Exemple

Transformer toutes les variables de type facteur en variable de type caractère.

```
h2 = hdv2003 %>% mutate_if(is.factor,as.character)
```

Compléments

Syntaxe dplyr

- Grammaire des données
- Philosophie reprise dans ggplot2
- Package structurant : tidyverse

Package tidyverse

- Composantes
  - → ggplot2() : Traitements graphiques
  - → tidyr() : Mise en forme de données
  - → dplyr() : Gestion de données
  - → readr() : Importation de données ASCII
  - → stringr() : Traitement chaînes de caractères
  - → forcats() : Gestion des facteurs
  - → readxl()* : Importation de données Excel
  - → purrr() : Programmation fonctionnelle
  - → tibble() : Définition objet tibbles
  - → lubridate()* : Traitement des dates

Tibbles

- Tibbles = extension data frames
  - →Pas de noms de lignes
  - →Noms de colonnes plus permissifs
  - →Affichage amélioré

Exemple d’utilisation de stringr

```
cn = cn %>%
        str_replace("Mihcel", "Michel") %>%
        str_replace("Loiuse", "Louise") %>%
        str_replace("Charnile", "Charline")
```

Opérations sur les métadonnées

- Renommer une série de variables

  →rename_at

- Recodification automatique

  →mutate_at

- Changement de type

  →cf mutate_if

Liens Web

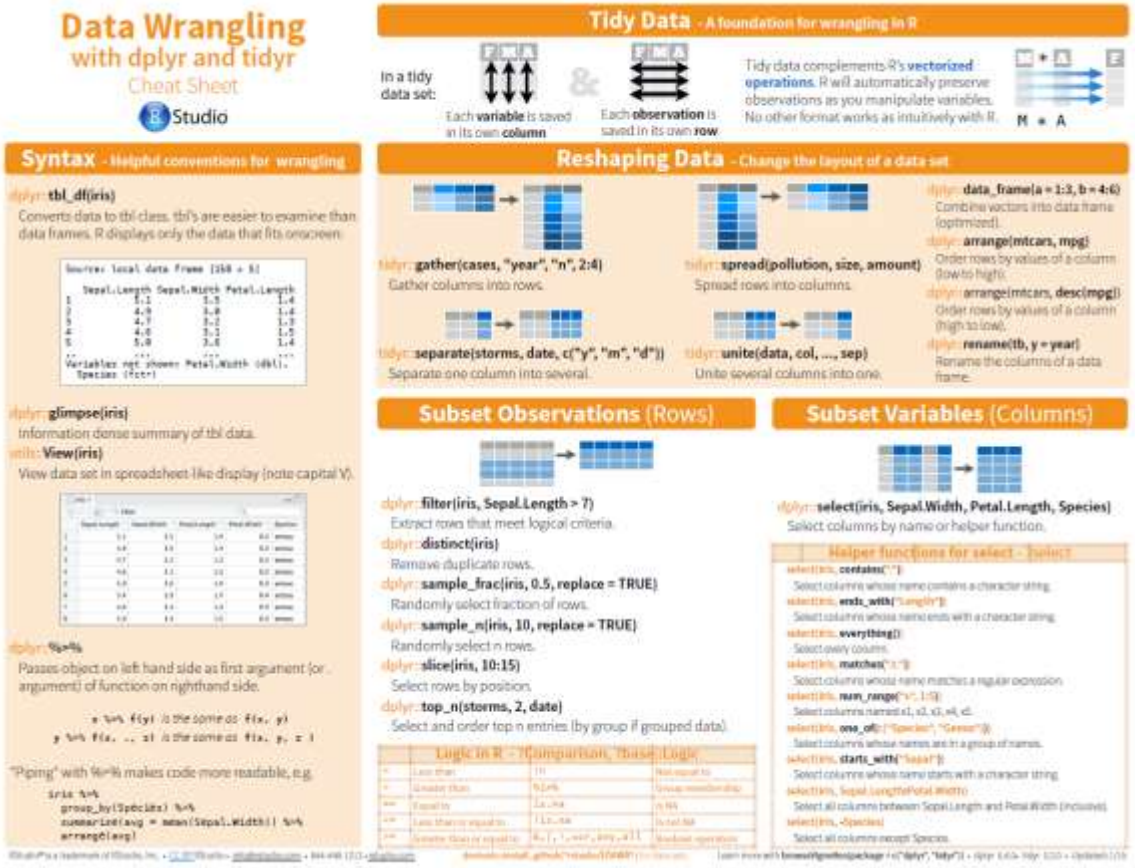
- Tidyverse :
  → https://www.tidyverse.org/

- Documentation J. Barnier :
  → https://juba.github.io/tidyverse/10-dplyr.html

- Wickham, Hadley. « Tidy Data ». *Journal of Statistical Software; Vol 1, Issue 10 (2014)*,
  → https://www.jstatsoft.org/v059/i10

Autres ressources

- Cheat sheets :



https://www.rstudio.com/resources/cheatsheets/