Économétrie 1
Introduction à la gestion de base de données : les packages dplyr et tidyr

Carla Morvan

L3 SEG

2023 - 2024

GATE Lyon / St-Etienne

Université Lumière Lyon 2

Introduction : Package dplyr et tidyr

Introduction à la gestion de bases de données (BDD) :

- Les packages dplyr et tidyr permettent de manipuler facilement des données avec R.
  - Vous souvenez-vous comment on installe un package ?
  - install.packages("dplyr")
  - library(dplyr) : à quoi sert cette commande déjà ?
- Ils permettent également le chaînage de commandes : %>%
  - Plutôt qu'une commande par ligne, on a une seule ligne pour plusieurs actions.
  - C'est déjà une maîtrise de R de plus haut niveau.
- Ces packages offrent plusieurs fonctions pour modifier les bases de données :
  - Sélectionner certaines lignes ou colonnes,
  - Créer de nouvelles variables à partir d'autres colonnes (avec plein d'arguments),
  - Trier les données, fusionner des données,
  - Reformater des données...

Sélectionner des observations :

La commande filter():

- La fonction filter() du package dplyr permet de sélectionner des observations vérifiant une ou plusieurs conditions.
- Syntaxe : filter(data, condition1, condition2, ...)
- **Application :** ▸ Code pour obtenir la BDD

Niveau 1 Utilisez la base de données "storms" et filtrez les données pour obtenir uniquement les tempêtes dont le vent était supérieur ou égal à 50 km/h.

Niveau 2 Filtrez les données pour obtenir uniquement les tempêtes dont le vent est supérieur ou égal à 50 km/h et ayant eu lieu après 2017.

Niveau 3 Filtrez les données et utilisez un chaînage de commandes pour obtenir uniquement les tempêtes dont le vent est supérieur ou égal à 50 km/h parmi (%in%) les tempêtes "Alex", "Xynthia" et "Jo".

Sélectionner des observations :

La commande select() :

- La fonction select() du package dplyr permet de conserver ou de supprimer des variables dans une table.
- Syntaxe : select(data, variable1, variable2...)

- **Application :**

Niveau 1 Utilisez la base de données "storms" et conservez uniquement les colonnes "storm" et "pressure".

Niveau 2 Supprimez uniquement la colonne "storm" (formulation la plus concise possible !).

Niveau 3 Conservez toutes les colonnes de la base qui commencent par "loc_" ainsi que les noms des tempêtes.

Renommer des variables :

La commande rename():

- La fonction rename() (et rename_with()) du package dplyr permet de renommer une ou plusieurs variables.
- Syntaxe : rename(data, NouveauNom = AncienNom)

- **Application :**

Niveau 1 Renommez la première colonne par un nom plus explicite.

Niveau 2 Changez les noms en anglais par des noms en français.

Niveau 3 Passez tous les noms des variables en majuscule.

Créer des nouvelles variables :

La commande mutate():

- La fonction mutate() du package dplyr permet de créer de nouvelles variables directement dans la base de données.
- Syntaxe : mutate(data, NomNouvelleVariable = ...)

- **Application :**

Niveau 1 Créez une nouvelle variable "ratio" qui représente le rapport pression de l'air / vitesse du vent.

Niveau 2 Créez une nouvelle variable qui calcule le logarithme du ratio.

Niveau 3 Créez une nouvelle variable qui combine les 3 variables de localisation pour former un code complet de la localisation de la tempête.

Créer des nouvelles variables :

La commande mutate():

- Dans la fonction mutate(), on peut intégrer des conditions logiques:
  - Une condition SI : ifelse(condition, valeurSIvrai, valeurSIfaux)
  - Une condition SI multiple : case_when(condition1 ~ valeur, condition2 ~ valeur....)

- **Application :**

Niveau 1 Créez une nouvelle variable catégorielle qui est égale à "FORT" si le vent est ≥ à 50 et "FAIBLE" sinon.

Niveau 2 Créez une nouvelle variable catégorielle qui est égale à "FORT" si le vent est ≥ à 50 et que la pression est inférieure à 1010, et "FAIBLE" sinon.

Niveau 3 Créez une nouvelle variable catégorielle qui est égale à "FORT" si le vent est ≥ à 70, "MOYEN" si le vent est compris entre 70 et 50, et "FAIBLE" sinon.

Créer des nouvelles variables :

La commande mutate():

- Dans la fonction mutate(), on peut intégrer des conditions logiques :
  - Une condition SI : ifelse(condition, valeurSIvrai, valeurSIfaux)
  - Une condition SI multiple : case_when(condition1 ~ valeur, condition2 ~ valeur....)

- **ATTENTION : Ne pas confondre mutate(+ condition) avec la commande mutate_if()**
- mutate_if() permet d'appliquer un traitement à certaines variables d'une base de données

- **Exemple :**
  - On divise par deux toutes les variables numériques de la base de données
  - mutate_if(storms, is.numeric, funs(./2))

Créer des nouvelles variables :

La commande mutate():

- Dans la fonction mutate(), on peut intégrer des conditions logiques :
  - Une condition SI : ifelse(condition, valeurSIvrai, valeurSIfaux)
  - Une condition SI multiple : case_when(condition1 ~ valeur, condition2 ~ valeur....)

- **ATTENTION : Ne pas confondre mutate(+ condition) avec la commande mutate_if()**
- mutate_if() permet d'appliquer un traitement à certaines variables d'une base de données

- **Exemple :**
  - On divise par deux toutes les variables numériques de la base de données
  - mutate_if(storms, is.numeric, funs(./2))

Trier les observations :

La commande arrange():

- La fonction arrange() du package dplyr permet de trier les observations par rapport à une ou plusieurs variables.
- Syntaxe : arrange(data, variable1, variable2...)

- **Application :**

Niveau 1 Triez la base par la vitesse du vent.

Niveau 2 Triez la base par la vitesse du vent et par date.

Niveau 3 Faites un tri décroissant de la vitesse du vent et croissant des dates.

Fusionner des bases de données :

Les commandes de fusion :

- Les fonctions left_join(), right_join(), inner_join(), full_join() du package dplyr permettent de fusionner deux bases de données en contrôlant la jointure des variables de fusion.
- Syntaxe : full_join(data1, data2, by=c())
- **Application :** ▸ Code pour obtenir la BDD

1. Fusionnez les bases 1 et 2 par la gauche.
2. Fusionnez les bases 1 et 2 par la droite.
3. Appliquez une jointure interne aux bases 1 et 2.
4. Appliquez une jointure totale aux bases 1 et 2.

Reformater des données avec gather():

- La fonction gather() du package tidyr permet de regrouper plusieurs colonnes en deux colonnes.
  - Cette fonction permet de mettre la base de données en format "Panel".
- Syntaxe : gather(data, colonne_Nom, colonne_Valeur, colonnes)

```
    country `2021` `2022` `2023`
1   FR         7000   6900   7000
2   DE         5800   6000   6200
3   GB        15000  14000  13000

> gather(pop, "year", "population", 2:4)
country year  population
1   FR      2021       7000
2   DE      2021       5800
3   GB      2021      15000
4   FR      2022       6900
5   DE      2022       6000
6   GB      2022      14000
7   FR      2023       7000
8   DE      2023       6200
9   GB      2023      13000
```

Reformater des données avec spread():

- La fonction spread() du package tidyr réalise l'inverse de gather().
  - Cette fonction permet de mettre la base de données au format "Panel".
- Syntaxe : spread(data, colonne_Nom, colonne_Valeur)

```
 city       polution niveau
1 Lyon        min           12
2 Lyon        max           58
3 Paris       min           17
4 Paris       max           38
5 Marseille   min            9
6 Marseille   max           39

> spread(pol, polution, niveau)

  city         max   min
1 Lyon          58    12
2 Marseille     39     9
3 Paris         38    17
```

Chaînage de commandes

- Un grand intérêt des packages dplyr et tidyr est qu'ils permettent le chaînage de commandes.
- Plutôt que de coder plusieurs lignes pour atteindre nos objectifs, nous pouvons le faire directement en une seule ligne.
- Syntaxe : data %>% commande1 %>% commande2 ...

- **Application :**

Niveau 1 Utilisez l'outil de chaînage de commandes pour conserver uniquement les observations en France.

Niveau 2 À partir de ces données uniquement françaises, créez une nouvelle variable indiquant si la tempête était avant ou après 2017.

Niveau 3 À partir de la base précédente, créez une nouvelle variable qui indique la moyenne du vent des tempêtes du NORD et des tempêtes du SUD.

Base de données STORMS

Back

```
### creation de la base de donnees STORMS
library(dplyr)

#####creations des differentes variables
#nom de la catastrophe
storm <- c("Roberto", "Alex", "Cindy", "Xynthia", "Fred", "Jo")
#vitesse du vent en km/h
wind <- c(110,45,65,120,50,45)
#pression atmospherique
pressure <- c(1007,1009,1005,1004,1010,1010)
#annee de la catastrophe
date <- c(2020,2018,2020,2015,2017,2014)
#localisation
loc_pays <- c("FR","FR","FR","FR","FR","GB")
loc_reg <- c(4,2,4,1,5,12)
loc_dep <- c(69,13,42,80,85,53)

### creation de la base de donnee
storms <- data_frame(storm, wind, pressure, date, loc_pays, loc_reg
    , loc_dep)

#verification de la structure de la base
str(storms)
```

Bases de données pour la fusion

Back

```
####fusionner des donnees####

##creation des bases de donnees pour l'exemple
A <- c("a","b","c")
B <- c("t","u","v")
C <- c("1","2","3")
data1 <- data_frame(A,B,C)

A <- c("a","b","d")
B <- c("t","u","w")
D <- c("3","2","1")
data2 <- data_frame(A,B,D)
```

Arguments des fonctions de dplyr

Syntaxe	Action
-	Dans la fonction select, sélectionne tout sauf
contains("a")	Le nom de la colonne contient la chaîne de caractères a
starts_with("a")	Le nom de la colonne commence par a
ends_with("a")	Le nom de la colonne se termine par a
toupper	Met en majuscule
tolower	Met en minuscule
substring(variable, 2)	Extrait une chaîne de caractères depuis la position 2
substring(variable, 2, 4)	Extrait depuis la position 2 jusqu'à la 4
paste(chaine1, chaine2)	Concaténation des deux chaînes de caractères
paste0(chaine1, chaine2)	Concaténation sans espaces
%>%	Permet de chaîner 2 commandes
%in%	A l'intérieur d'une commande, permet d'exclure des obs
group_by()	Permet de définir des groupes