Chapitre 5 Première manipulation des données

5.1 Afficher les valeurs et manipuler les variables

- Pour afficher la table, plusieurs façon : “clic” dans l’environnement Rstudio, `View(base)` , `print(base)` , `base` .
- Pour accéder à une variable : fonction `pull`

Par exemple :

```
library (tidyverse)
base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",")
str (pull (base, DEP))
```

```
##  Factor w/ 100 levels "01","02","03",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- Création de nouvelles variables : fonction `mutate`

```
TableEnSortie <- mutate (TableEnEntree,
                         NouvelleVariable = DefinitionDeLaVariable)
```

Par exemple :

```
base <- mutate (base, log_SUPERF = log (SUPERF))
```

Nb : `mutate` permet également de modifier une variable. Dans ce cas la syntaxe est la même que ci-dessus, mais les noms d’entrée et de sortie sont les mêmes

```
base <- mutate (base, log_SUPERF = 100 * log_SUPERF)
```

⇒ La table base contient de nouvelles colonnes

5.2 Sélectionner des variables

La fonction `select`

- sélection par liste blanche

```
TableEnSortie <- select (TableEnEntree, Variable1, Variable2, ..., Va
```

- sélection par liste noire (supprimer)

```
TableEnSortie <- select (TableEnEntree, -Variable1, -Variable2, ...,
```

Par exemple :

```
base_select <- select (base, CODGEO, LIBGEO, P14_POP)
base_select <- select (base, -CODGEO)
```

5.3 Sélectionner des observations

La fonction `filter`

```
TableEnSortie <- filter (TableEnSortie, Condition1, ..., ConditionN)
```

Par exemple :

```
base_filter <- filter (base, DEP == "01" & P14_POP > 10000)
```

⇒ Attention à l'opérateur de comparaison : " == " et non pas "="

5.4 Les tests logiques dans R

Syntaxe	Action
==	Test d'égalité
!=	Différent de
%in% c(…)	Dans une liste de valeurs
$>, >= <, <=$	Supérieur (ou inférieur) (ou égal)
! (x %in% c(…))	N'est pas dans une liste de valeurs
x==a & y==b	x vaut a **ET** y vaut b
x==a	y==b

5.5 Renommer des colonnes

Pour renommer des variables d'un dataframe, on utilise la fonction `rename` qui prend la syntaxe générale suivante :

```
base <- rename (base, nouveau_nom = ancien_nom)
```

Exemple

```
base_rename <- rename (base, ZONE_EMPLOI = ZE)
```

5.6 Exercice : Créer, filtrer, sélectionner

- En utilisant la fonction `mutate()` , créer une nouvelle variable correspondant à la densité de population (rapport de la population à la superficie de la commune), ainsi que les taux de natalité et de mortalité (en pour mille)
- A l'aide de la fonction `select()` , créer une nouvelle table en ne conservant que le code commune, le type de commune (ZAU), la région, le département et les variables que vous venez de créer.
- Enfin, ne conserver les communes correspondant à votre département de naissance et stocker ce *dataframe*. Attention au type de la variable département !
- Avec les opérateurs logiques, faire des essais pour sélectionner des échantillons différents.

```
df <- mutate (base, densite = P14_POP / SUPERF,
              tx_natal = 1000 * NAISD15 / P14_POP,
              tx_mort = DECESD15 / P14_POP)


selection <- select (df, CODGEO, ZAU, REG, DEP,
                     densite, tx_natal, tx_mort)


S0 <- filter (selection, DEP == "62")
```

```
S1 <- filter (selection, DEP != "62") # tout sauf le 62 :(
S2 <- filter (selection, DEP %in% c ("59","62")) # L'ancien NPdC :)
S3 <- filter (selection, !(DEP %in% c ("59","62"))) # Le "sud" de la
S4 <- filter (selection, densite > 100) # l'urbain
S5 <- filter (selection, DEP=="62" & densite > 100) # le PdC urbain
S6 <- filter (selection, DEP=="62" | densite > 100) # le PdC et l'urb
```

Nom d'un pipe `%>%` !

On peut combiner les opérations en une seule ligne à l'aide du pipe `%>%` :

```
selection_62 <- select (df, CODGEO, ZAU, REG, DEP, densite, tx_natal)
             filter(DEP=="62")
```

Le coin du capitaine [ ] 

Pour sélectionner les données, on peut également utiliser les crochets couplés aux dollars et aux `c()` , sans passer par les numéros de lignes et colonnes ! On rencontre assez souvent cette syntaxe sur les forums (voir chapitre 9).

```
selection_62 <- df [df$DEP == "62",
                    c ("CODGEO", "ZAU", "REG", "DEP", "densite",
                       "tx_natal")]
```

Note : dans la syntaxe `tidyverse` , on ne met pas (obligatoirement) de guillemets pour les noms de variable, alors qu'on le fait pour la version [ ]