# Chapitre 6: les liaisons de données

*Traduit par Marie Delacre.*

## 6.1 Objectifs d’apprentissage

### 6.1.1 Niveau débutant

1. Être capable d’utiliser les 4 fonctions de jointure :

o [left\_join()](#_3d5qvurzv9g0)

o [right\_join()](#_fu1zqm3zf7cb)

o [inner\_join()](#_6z108k48ch4u)

o [full\_join()](#_cvh73peyoox7)

2. Utiliser l’argument *by* pour définir les colonnes servant à joindre les tables

### 6.1.2 Niveau intermédiaire

3. Utiliser l’argument *suffix* pour discriminer plusieurs colonnes qui portent le même nom

4. Être capable d’utiliser les 2 fonctions de jointures filtrantes :

o [semi\_join()](#_o33eiy9fp36e)

o [anti\_join()](#_7rnmi1syxjnb)

5. Être capable d’utiliser les 2 fonctions de concaténation de tableaux :

o [bind\_rows()](#_hkbx9p1fli39)

o [bind\_cols](#_jk6048hw6zto)[()](#_7rnmi1syxjnb)

6. Être capable d’utiliser les 3 opérations ensemblistes :

o [intersect()](#_xojdz33klwpd)

o [union()](#_has8rrhjtd58)

o [setdiff()](#_i2dwoqig8cxo)

## 6.2 Resources

·Le chapitre 13 [Relational Data](https://r4ds.had.co.nz/relational-data.html) de [*R for Data Science*](https://r4ds.had.co.nz/) (EN)

·[L’antisèche pour les fonctions de jointure de deux tables, du package dplyr](https://stat545.com/join-cheatsheet.html) (EN)

·[Slides du cours sur les fonctions de jointures de deux tables, du package dplyr](https://psyteachr.github.io/msc-data-skills/slides/05_joins_slides.pdf) (EN)

## 

## 6.3 Les données

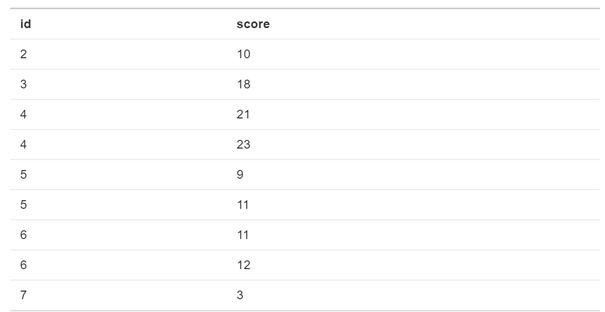
Nous allons commencer par créer deux petites tables de données.

Dans la table subject, on trouve l’identifiant (id), le sexe (les hommes sont codés « m » et les femmes sont codées « f ») et l’âge de 5 sujets, à l’exception de l’âge et du sexe du sujet 3 pour lequel ces données sont manquantes.

|  |
| --- |
| subject <- **tibble**(  id = **seq**(1,5),  sex = **c**("m", "m", NA, "f", "f"),  age = **c**(19,22,NA,19,18)  ) |

Dans la table exp, on trouve l’identifiant (id) de sujets ainsi que le score qu’ils ont obtenu à une expérience. Certains sujets de la table subject sont manquants, d’autres ont réalisé deux fois l’expérience et enfin, certains sujets répertoriés dans la table exp ne sont pas mentionnés dans la table subject.

|  |
| --- |
| exp <- **tibble**(  id = **c**(2, 3, 4, 4, 5, 5, 6, 6, 7),  score = **c**(10, 18, 21, 23, 9, 11, 11, 12, 3)  ) |



## 6.4 Les jointures

Toutes les fonctions de jointures ont la syntaxe de base suivante :

\*\*\*\*\_join(x, y, by = NULL, suffix = c(".x", ".y")

· x = la première table (table de gauche)

· y = la deuxième table (table de droite)

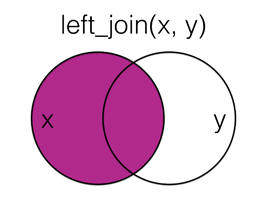
· {#join-by} by = quelles sont les colonnes qui permettent de joindre les deux tables? Si vous laissez ce champ vide, la jointure sera faite sur base de toutes les colonnes qui ont le même nom dans les deux tables.

· {#join-suffix} suffix = si des colonnes ont le même nom dans les deux tables mais que vous ne les utilisez pas pour la jointure, un suffixe sera ajouté à leur nom dans la table finale, de sorte à les rendre non ambigües. Par défaut, les suffixes ajoutés sont “.x” et “.y”, mais vous pouvez les remplacer par quelque chose de plus pertinent.



|  |
| --- |
| Bien que l’argument “by” puisse être omis si vous faites la jointure sur base de toutes les colonnes ayant le même nom dans les deux tables, il est recommandé de systématiquement spécifier cet argument, de sorte que votre code soit robuste  aux modifications dans les données chargées. |

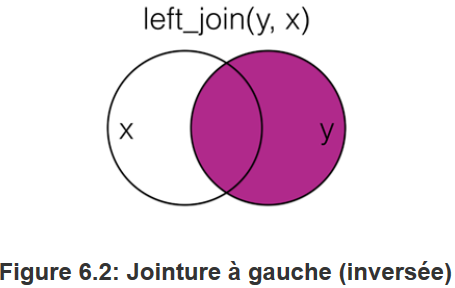
### 6.4.1 Jointure à gauche: left\_join()

**Figure 6.1: Jointure à gauche**

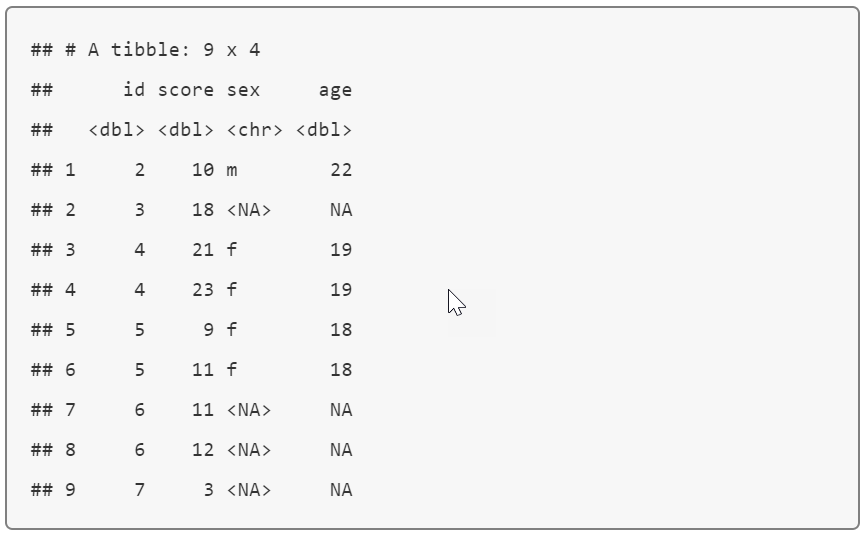
La fonction *left\_join* conserve toutes les données de la première table (à gauche) auxquelles elle adjoint les données provenant de la deuxième table (à droite). S’il y a plusieurs lignes de la table de droite en correspondance avec une ligne de la table de gauche, cette dernière sera dupliquée dans la table jointe (voir les sujets 4 et 5).

|  |
| --- |
| **left\_join**(subject, exp,by = "id") |



Dans le code ci-dessous, nous avons inversé l’ordre des tables dans la fonction, de sorte que la table jointe résultante soit l’ensemble des lignes de la table exp auxquelles sont adjointes les informations provenant de la table subject.

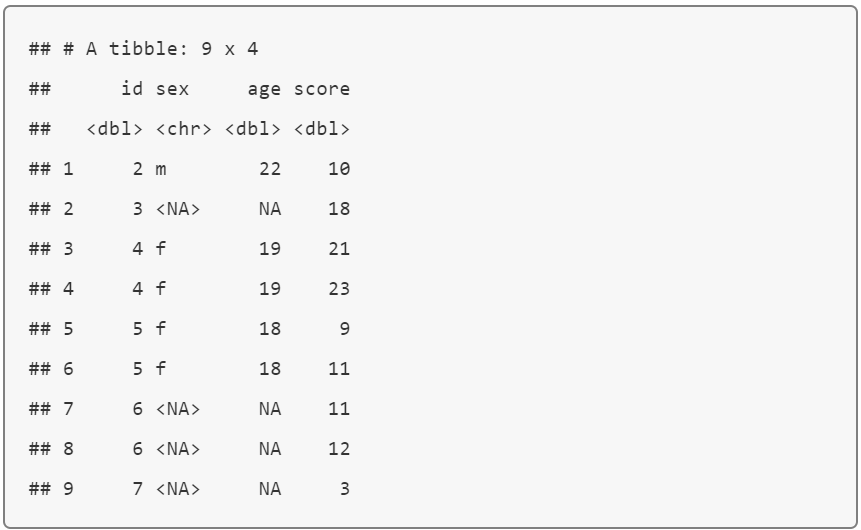
|  |
| --- |
| **left\_join**(exp, subject,by = "id") |

****

### 6.4.2 Jointure à droite: right\_join()

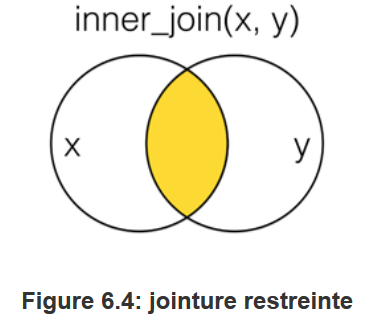
La fonction *right\_join* conserve toutes les données de la deuxième table (à droite) auxquelles elle adjoint les données provenant de la première table (à gauche).

|  |
| --- |
| **right\_join**(subject, exp,by = "id") |

****



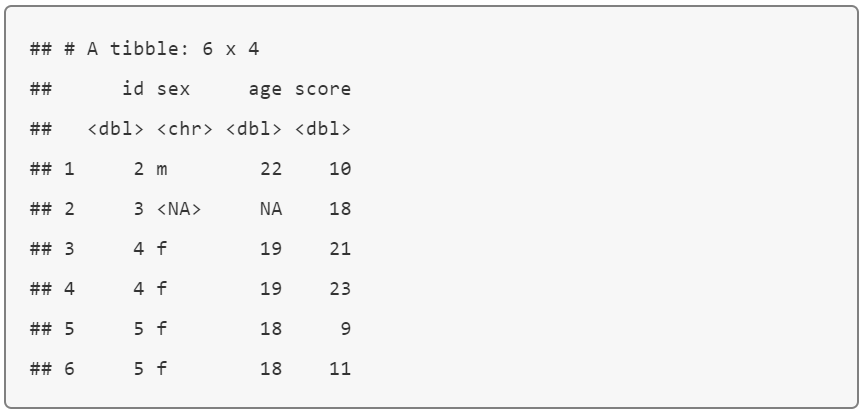
|  |
| --- |
| Cette table contient la même information que celle obtenue via la fonction *left\_join(exp, subject, by = =”id”)*, mais les colonnes sont présentées dans un ordre différent (les colonnes de la table de gauche précèdent celles de la table de  droite) |



### 6.4.3 Jointure restreinte: inner\_join()

La fonction *inner\_join* produira une table jointe constituée de toutes les lignes qui sont présentes dans les deux tables simultanément.

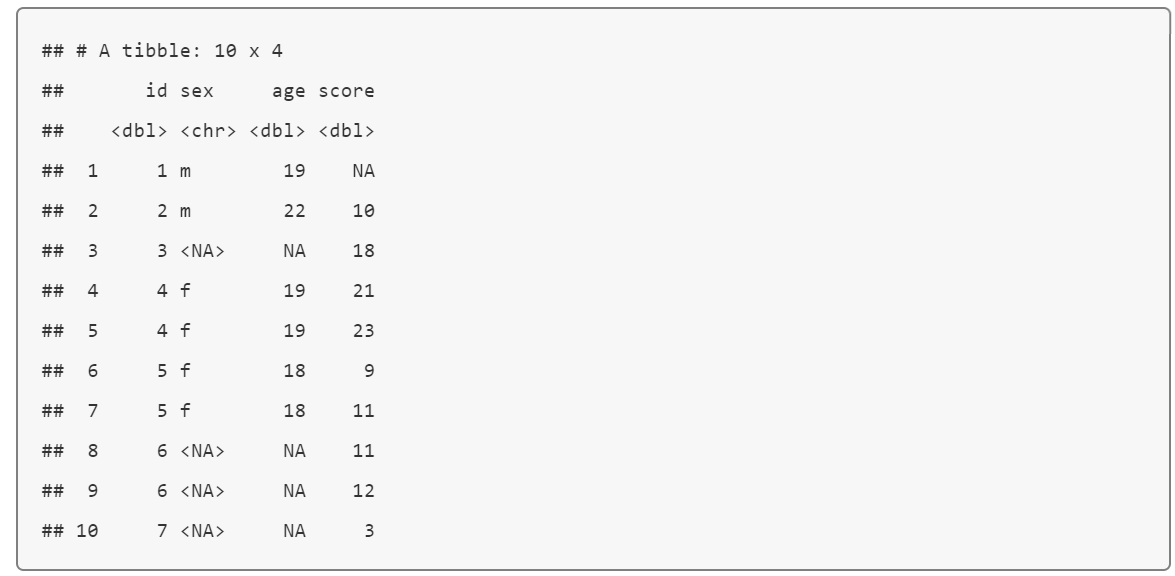
|  |
| --- |
| **inner\_join**(subject, exp,by = "id") |

****

### 6.4.4 Jointure complète: full\_join()

La fonction *full\_join* vous permet de rejoindre les lignes des deux tables tout en conservant l’ensemble de l’information des deux tables. Si une ligne n’est pas présente dans les deux tables simultanément, les valeurs « NA » seront introduites dans les colonnes correspondant à la table dans laquelle la ligne n’est pas présente.

|  |
| --- |
| **full\_join**(subject, exp,by = "id") |

****

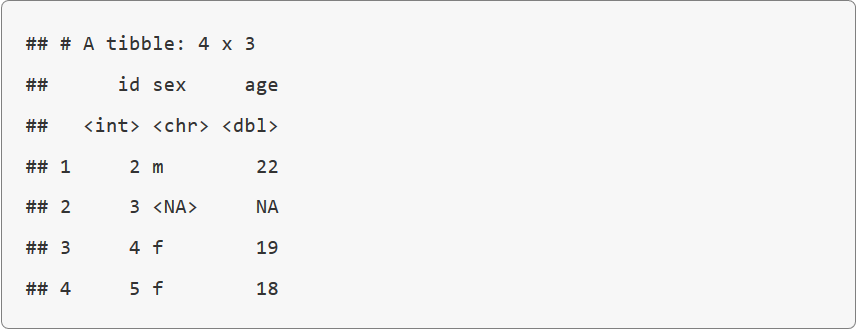
## 6.5 Jointures filtrantes

### 6.5.1 Semi-jointure : semi\_join()

**Figure 6.6: semi jointure**

La fonction *semi\_join* sélectionne toutes les lignes de la table de gauche qui ont une correspondance dans la table de droite, sans ajouter les colonnes de la table de droite.

|  |
| --- |
| **semi\_join**(subject, exp, by = "id") |

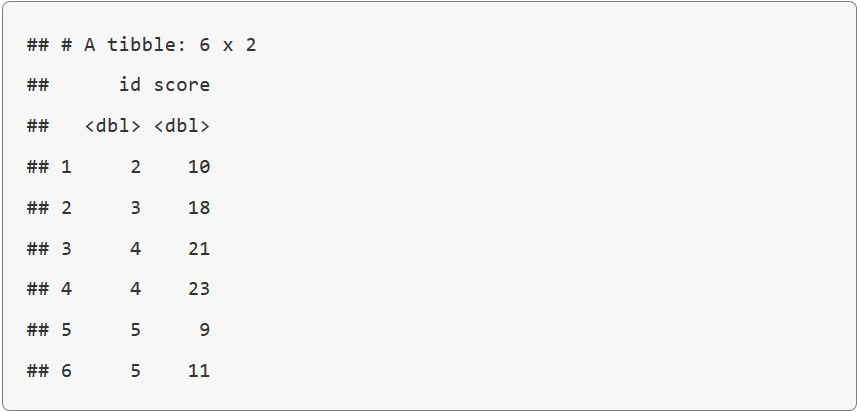
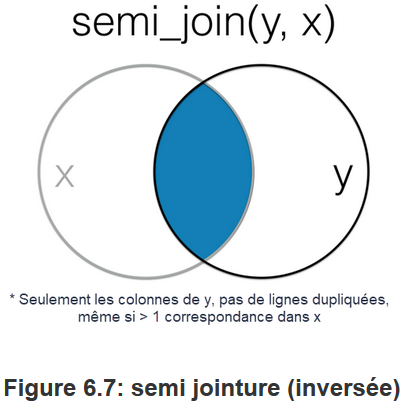
****



|  |
| --- |
| Contrairement à la fonction *inner\_join*, *semi\_join* ne dupliquera jamais les lignes de la table de gauche s’il y a plusieurs lignes qui lui correspondent dans la table de droite. |

L’ordre dans lequel on introduit les tables dans la fonction *semi\_join* importe.

|  |
| --- |
| **semi\_join**(exp, subject, by = "id") |

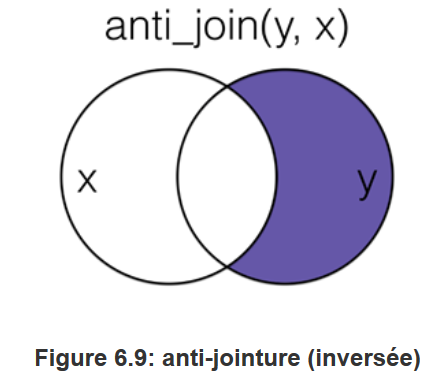
****

### 6.5.2 Anti-jointure : anti\_join()

La fonction *anti\_join* sélectionne toutes les lignes de la table de gauche qui *n’*ont pas de correspondance dans la table de droite, sans ajouter les colonnes de la table de droite.

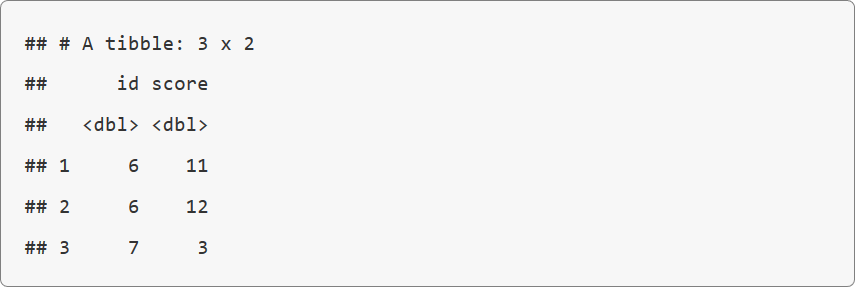
|  |
| --- |
| **anti\_join**(subject, exp, by = "id") |





L’ordre dans lequel on introduit les tables dans la fonction *anti\_join* importe.

|  |
| --- |
| **anti\_join**(exp, subject, by = "id") |

****

## 6.6 Les fonctions de concaténations

### 6.6.1 Concaténation de lignes: bind\_rows()

Vous pouvez combiner les lignes de deux tables avec la fonction *bind\_rows*.

Dans l’exemple ci-dessous, nous allons créer la table « new\_subjects » contenant l’identifiant (id), le sexe et l’âge des sujets 6 à 9, et concaténer cette table et la table d’origine « subject ».

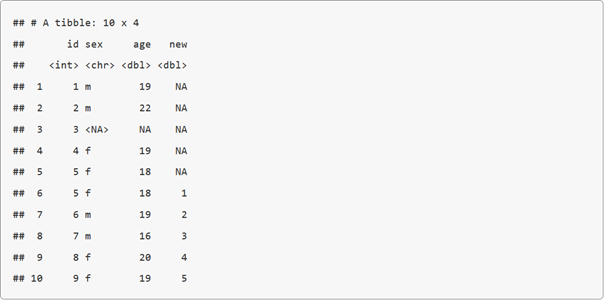
|  |
| --- |
| new\_subjects <- **tibble**(  id = **seq**(6, 9),  sex = **c**("m", "m", "f", "f"),  age = **c**(19, 16, 20, 19)  )  **bind\_rows**(subject, new\_subjects) |

Les colonnes qui correspondent à une variable doivent avoir le même nom d’une table à l’autre pour pouvoir être associées, mais il n’est pas nécessaire qu’elles soient présentées dans le même ordre d’une table à l’autre.

Si certaines colonnes sont manquantes dans une des deux tables, des NA seront automatiquement insérés.

Si une ligne est présente dans les deux tables (comme la ligne qui se rapporte à l’id 5, présente à la fois dans les tables **subjects** et **new\_subjects** ci-dessous), alors, elle sera dupliquée dans la table finale.

|  |
| --- |
| new\_subjects <- **tibble**(  id = **seq**(5, 9),  age = **c**(18, 19, 16, 20, 19),  sex = **c**("f", "m", "m", "f", "f"),  new = **c**(1,2,3,4,5)  )  **bind\_rows**(subject, new\_subjects) |

****

Si vos tables contiennent exactement les mêmes colonnes, vous pouvez utiliser la fonction *union()* (voir ci-dessous) pour éviter les duplications de lignes.

### 6.6.2 Concaténation de colonnes: bind\_cols()

Vous pouvez fusionner deux tables ayant le même nombre de lignes en utilisant la fonction *bind\_cols*. Ce n’est utile que si les lignes sont présentées exactement dans le même ordre dans les deux tables. Le seul avantage de cette fonction, par rapport à la jointure à gauche, survient lorsque les tables n’ont pas de colonne « id » permettant de joindre l’information des deux tables, nous obligeant à nous fier uniquement à leur ordre.

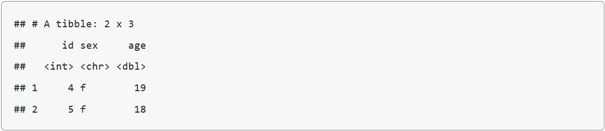
|  |
| --- |
| new\_info <- **tibble**(  colour = **c**("red", "orange", "yellow", "green", "blue")  )    **bind\_cols**(subject, new\_info) |

## 6.7 les opérations ensemblistes

### 6.7.1 la fonction *intersect()*

La fonction *intersect()* sélectionne toutes les lignes qui sont en parfaite correspondance (c’est-à-dire, les lignes qui contiennent exactement les mêmes donnés, relatives aux mêmes variables) dans les deux tables, peu importe si les lignes sont présentées dans le même ordre ou non.

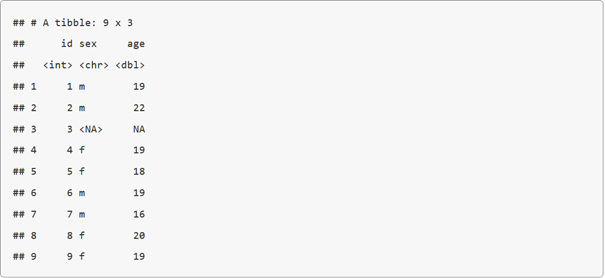
|  |
| --- |
| new\_subjects <- **tibble**(  id = **seq**(4, 9),  age = **c**(19, 18, 19, 16, 20, 19),  sex = **c**("f", "f", "m", "m", "f", "f")  )    dplyr::**intersect**(subject, new\_subjects) |

****

### 6.7.2 la fonction *union()*

La fonction *union()* fusionne toutes les lignes des deux tables, en supprimant les doublons (à condition que les deux tables contiennent exactement les mêmes colonnes, comme il a été mentionné précédemment).

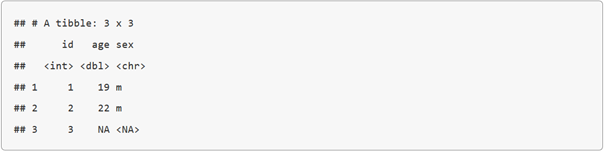
|  |
| --- |
| dplyr::**union**(subject, new\_subjects) |

****

### 6.7.3 la fonction *setdiff()*

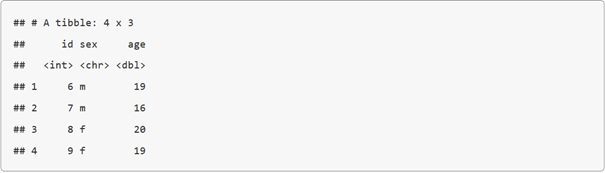
La fonction *setdiff* sélectionne les lignes qui ne sont présentes que dans la première table (et qui sont donc absentes de la seconde table).

|  |
| --- |
| **setdiff**(subject, new\_subjects) |

****

L’ordre dans lequel on introduit les tables dans la fonction *setdiff* importe.

|  |
| --- |
| **setdiff**(new\_subjects, subject) |

****

## 6.8 Exercices

Téléchargez les [exercices](https://psyteachr.github.io/msc-data-skills/exercises/06_joins_exercise.Rmd). Ne regardez les [réponses](https://psyteachr.github.io/msc-data-skills/exercises/06_joins_answers.Rmd) qu’après avoir tenté de répondre à toutes les questions