Chapitre 4 : Données *tidy*

*Traduit par* [*Ladislas Nalborczyk*](https://www.barelysignificant.com)*.*

Pour savoir si vous avez besoin de réviser ce chapitre, faites le **quiz** (voir 4.8).

# 4.1 Objectifs d'apprentissage

## 4.1.1 Niveau débutant

1. Comprendre le concept de données *tidy* (“bien rangées”)
2. Être capable d’utiliser les 4 verbes fondamentaux de tidyr:
   * gather()
   * separate()
   * spread()
   * unite()
3. Être capable d’enchaîner les fonctions en utilisant les *pipes*

## 4.1.2 Niveau intermédiaire

1. Être capable d’utiliser des arguments comme sep, extra, et convert, afin de manipuler des jeux de données plus compliqués

## 4.1.3 Niveau avancé

1. Être capable d’utiliser des [expressions régulières](https://fr.wikipedia.org/wiki/Expression_r%C3%A9guli%C3%A8re) afin de séparer des colonnes complexes

# 4.2 Ressources

*N.B. Les ressources marquées (EN) sont en anglais.*

* [Tidy Data](http://vita.had.co.nz/papers/tidy-data.html) (EN)
* [Chapter 12: Tidy Data](http://r4ds.had.co.nz/tidy-data.html) de *R for Data Science (EN)*
* [Chapter 18: Pipes](http://r4ds.had.co.nz/pipes.html) de *R for Data Science (EN)*
* [Data wrangling cheat sheet](https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) (EN)

# 4.3 Configuration

|  |
| --- |
| *# libraries needed*  **library**(tidyverse)  **library**(readxl)  **set.seed**(8675309) *# make sure random numbers are reproducible* |

# 4.4 Trois règles pour des données *tidy*

# Chaque [variable](https://psyteachr.github.io/glossary/v#variable) doit avoir sa propre colonne

# Chaque [observation](https://psyteachr.github.io/glossary/o#observation) doit avoir sa propre ligne

# Chaque [valeur](https://psyteachr.github.io/glossary/v#value) doit avoir sa propre cellule

# Le tableau ci-dessous contient trois observations par ligne. La colonne total\_meanRT contient deux valeurs.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **score\_1** | **score\_2** | **score\_3** | **rt\_1** | **rt\_2** | **rt\_3** | **total\_meanRT** |
| 1 | 4 | 3 | 7 | 857 | 890 | 859 | 14 (869) |
| 2 | 3 | 1 | 1 | 902 | 900 | 959 | 5 (920) |
| 3 | 2 | 5 | 4 | 757 | 823 | 901 | 11 (827) |
| 4 | 6 | 2 | 6 | 844 | 788 | 624 | 14 (752) |
| 5 | 1 | 7 | 2 | 659 | 764 | 690 | 10 (704) |

Ci-dessous la version *tidy* (“bien rangée”).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **trial** | **rt** | **score** | **total** | **mean\_rt** |
| 1 | 1 | 857 | 4 | 14 | 869 |
| 1 | 2 | 890 | 3 | 14 | 869 |
| 1 | 3 | 859 | 7 | 14 | 869 |
| 2 | 1 | 902 | 3 | 5 | 920 |
| 2 | 2 | 900 | 1 | 5 | 920 |
| 2 | 3 | 959 | 1 | 5 | 920 |
| 3 | 1 | 757 | 2 | 11 | 827 |
| 3 | 2 | 823 | 5 | 11 | 827 |
| 3 | 3 | 901 | 4 | 11 | 827 |
| 4 | 1 | 844 | 6 | 14 | 752 |
| 4 | 2 | 788 | 2 | 14 | 752 |
| 4 | 3 | 624 | 6 | 14 | 752 |
| 5 | 1 | 659 | 1 | 10 | 704 |
| 5 | 2 | 764 | 7 | 10 | 704 |
| 5 | 3 | 690 | 2 | 10 | 704 |

# 4.5 Organiser ses données

Télécharger les données contenues dans le fichier [personality.csv](https://psyteachr.github.io/msc-data-skills/data/personality.csv). Ces données sont issues d’un questionnaire célèbre de personnalité à 5 facteurs (OCEAN). Chaque question est labellisée par la dimension à laquelle elle appartient (Op = ouverture (*openness*), Co = conscience professionnelle (*conscientiousness*), Ex = extraversion, Ag = agréabilité (*agreeableness*), and Ne = névrosisme (*neuroticism*)) ainsi que par le numéro de question.

|  |
| --- |
| *ocean <-* **read\_csv***("https://psyteachr.github.io/msc-data-skills/data/personality.csv")* |

<< the resulting table seems too large to be copy-pasted here…>

## 4.5.1 gather()

gather(data, key = "key", value = "value", ..., na.rm = FALSE, convert = FALSE, factor\_key = FALSE)

* key détermine le nom de la colonne qui contiendra les noms de lignes (“question” dans notre exemple).
* value détermine le nom de la colonne qui contiendra les valeurs contenues dans les colonnes que l’on souhaite rassembler (“score” dans cet exemple).
* ... fait référence aux colonnes que l’on souhaite rassembler. On peut faire référence à ces colonnes soit par leur nom (e.g., col1, col2, col3, col4 ou col1:col4), soit par leur position (e.g., 8, 9, 10 ou 8:10).
* na.rm détermine si les lignes qui contiennent des valeurs manquantes (NA) doivent être supprimées.
* convert détermine si le résultat (ou plutôt les colonnes résultant) de l’appel à la fonction gather() doit être converti en un autre type de données que le type de données (des colonnes) d’origine.
* factor\_key détermine si les valeurs key (i.e., les labels) doivent être stockées comme un facteur (avec le même ordre que dans le tableau d’origine) ou comme un vecteur de caractères.

ocean est présenté au format large (*wide*), avec une colonne unique par question. Nous allons modifier ce jeu de données de manière à obtenir une ligne pour chaque observation (*user* \* *question*). Le tableau de données final doit contenir les colonnes suivantes: user\_id, date, question, et score.

|  |
| --- |
| ocean\_gathered <- **gather**(ocean, "question", "score", Op1:Ex9) |

|  |  |  |  |
| --- | --- | --- | --- |
| **user\_id** | **date** | **question** | **score** |
| 0 | 2006-03-23 | Op1 | 3 |
| 1 | 2006-02-08 | Op1 | 6 |
| 2 | 2005-10-24 | Op1 | 6 |
| 5 | 2005-12-07 | Op1 | 6 |
| 8 | 2006-07-27 | Op1 | 6 |
| 108 | 2006-02-28 | Op1 | 3 |

## 4.5.2 separate()

separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE, convert = FALSE, extra = "warn", fill = "warn")

* col est la colonne que nous souhaitons séparer
* into est un vecteur contenant le nom des nouvelles colonnes à créer
* sep est le caractère (ou les caractères) permettant de séparer la colonne d’origine. Par défaut, cela correspond à n’importe quel caractère qui n’est pas [alphanumérique](https://fr.wikipedia.org/wiki/Caract%C3%A8re_alphanum%C3%A9rique) (e.g., ,\_-/:)
* remove détermine si la colonne séparée (col) doit être supprimée du nouveau tableau de données. Par défaut, la colonne est supprimée.
* convert détermine si le résultat (ou plutôt les colonnes résultant) de l’appel à la fonction separate() doit être converti en un autre type de données que le type de données (des colonnes) d’origine.
* extra détermine ce qu’il doit advenir des colonnes supplémentaires (par exemple si trois colonnes résultent de la séparation alors que l’utilisateur n’a défini que deux noms de colonnes).
* fill détermine ce qui doit advenir des colonnes manquantes (par exemple si deux colonnes résultent de la séparation alors que l’utilisateur a défini trois noms de colonnes).

Nous allons diviser la colonne question en deux nouvelles colonnes: domain et qnumber.

Il n’y a pas de caractère à utiliser pour séparer la colonne dans cet exemple, mais nous pouvons séparer une colonne après un nombre spécifique de caractères en utilisant l’argument sep et en spécifiant un nombre entier. Par exemple, pour séparer le chaîne de caractères “abcde” après le troisième caractère, on peut utiliser sep = 3, qui donne le résultat suivant: c("abc", "de"). On peut aussi utiliser un nombre entier négatif, afin de séparer la colonne avant le nième caractère à partir de la droite. Par exemple, pour séparer une colonne qui contient deux mots de longueur variable et un suffixe composé de deux chiffres (e.g., “lisa03”, ”amanda38”), on peut utiliser sep = -2.

|  |
| --- |
| ocean\_sep <- **separate**(ocean\_gathered, question, **c**("domain", "qnumber"), sep = 2) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **user\_id** | **date** | **domain** | **qnumber** | **score** |
| 0 | 2006-03-23 | Op | 1 | 3 |
| 1 | 2006-02-08 | Op | 1 | 6 |
| 2 | 2005-10-24 | Op | 1 | 6 |
| 5 | 2005-12-07 | Op | 1 | 6 |
| 8 | 2006-07-27 | Op | 1 | 6 |
| 108 | 2006-02-28 | Op | 1 | 3 |

|  |  |
| --- | --- |
|  | Si vous souhaitez séparer une colonne à un point (“.”), vous devez utiliser sep = "\\.", et non pas sep = ".". Les deux barres obliques inversées (*backslashes*) permettent d’**échapper** le point final, ce qui le rend interprétable littéralement comme un point final, au lieu d’être interprété comme une expression régulière qui signifie “n’importe quel caractère”. |

## 4.5.3 unite()

unite(data, col, ..., sep = "\_", remove = TRUE)

* col est le nom de la nouvelle colonne unie
* ... fait référence aux colonnes que l’on souhaite réunir
* sep fait référence au(x) caractère(s) que l’on souhaite utilise pour séparer le contenu des colonnes réunies
* remove détermine si les colonnes réunies (...) doivent être supprimées du nouveau tableau de données. Par défaut, ces colonnes sont supprimées.

Nous allons réunir les colonnes domain et qnumber dans une nouvelle colonne dénommée domain\_n, au format suivant: “Op\_Q1”.

|  |
| --- |
| ocean\_unite <- **unite**(ocean\_sep, "domain\_n", domain, qnumber, sep = "\_Q") |

|  |  |  |  |
| --- | --- | --- | --- |
| **user\_id** | **date** | **domain\_n** | **score** |
| 0 | 2006-03-23 | Op\_Q1 | 3 |
| 1 | 2006-02-08 | Op\_Q1 | 6 |
| 2 | 2005-10-24 | Op\_Q1 | 6 |
| 5 | 2005-12-07 | Op\_Q1 | 6 |
| 8 | 2006-07-27 | Op\_Q1 | 6 |
| 108 | 2006-02-28 | Op\_Q1 | 3 |

## 4.5.4 spread()

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

On peut également inverser la procédure présentée ci-dessus. Par exemple, on peut convertir notre tableau de données du format long au format large (*long-to-wide*) en utilisant la fonction spread().

* key fait référence à la colonne qui contient le nom des nouveaux noms de colonnes
* value fait référence à la colonne qui contient les valeurs contenues dans les nouvelles colonnes au format large

|  |
| --- |
| ocean\_spread <- **spread**(ocean\_unite, domain\_n, score) |

<< the resulting table seems too large to be copy-pasted here…>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_id** | **date** | **Ag\_Q1** | **Ag\_Q2** | **Ag\_Q3** | **Ag\_Q4** | **Ag\_Q5** | **Ag\_Q6** | **Ag\_Q7** | **Co\_Q1** |
| 0 | 2006-03-23 | 2 | 1 | 1 | 1 | 3 | NA | 3 | 3 |
| 1 | 2006-02-08 | 0 | 6 | 0 | 0 | 6 | 6 | 0 | 0 |
| 2 | 2005-10-24 | 0 | 6 | 1 | 1 | 5 | 5 | 1 | 0 |
| 5 | 2005-12-07 | 4 | 0 | 1 | 4 | 0 | 2 | 1 | 3 |
| 8 | 2006-07-27 | 6 | 5 | 0 | 6 | 2 | 3 | 1 | 5 |
| 108 | 2006-02-28 | 5 | 4 | 4 | 5 | 5 | 4 | 3 | 4 |

# 4.6 Pipes



Les *pipes* offrent une manière élégante d’arranger le code dans un format plus lisible que le format classique.

Par exemple, imaginons que nous nous disposions d’un tableau de données avec 10 identifiants de participants, deux colonnes représentant les deux modalités A1 et A2 d’une variable A et deux colonnes représentant les deux modalités B1 et B2 d’une variable B. On voudrait calculer la moyenne des deux variables A (A1 et A2) ainsi que la moyenne des deux variables B (B1 et B2), et obtenir au final un tableau avec 10 lignes (une ligne par participant) et trois colonnes (id, A\_mean et B\_mean).

Une première manière de réaliser cette opération serait de simplement créer un nouvel objet à chaque étape et d’utiliser l’objet ainsi obtenu pour l’étape suivante. Cette manière de faire est plutôt simple à suivre mais nous aurons créé 6 objets non indispensables dans notre environnement. Ainsi, cette manière de faire peut devenir confuse lorsque utilisée au sein de longs scripts.

|  |
| --- |
| # make a data table with 10 subjects  data\_original <- **tibble**(  id = 1:10,  A1 = **rnorm**(10, 0),  A2 = **rnorm**(10, 1),  B1 = **rnorm**(10, 2),  B2 = **rnorm**(10, 3)  )  # gather columns A1 to B2 into "variable" and "value" columns  data\_gathered <- **gather**(data\_original, variable, value, A1:B2)  # separate the variable column at the \_ into "var" and "var\_n" columns  data\_separated <- **separate**(data\_gathered, variable, c("var", "var\_n"), sep = 1)  # group the data by id and var  data\_grouped <- **group\_by**(data\_separated, id, var)  # calculate the mean value for each id/var  data\_summarised <- **summarise**(data\_grouped, mean = **mean**(value))  # spread the mean column into A and B columns  data\_spread <- **spread**(data\_summarised, var, mean)  # rename A and B to A\_mean and B\_mean  data <- **rename**(data\_spread, A\_mean = A, B\_mean = B)  data |

|  |
| --- |
| ## # A tibble: 10 x 3  ## # Groups: id [10]  ## id A\_mean B\_mean  ## <int> <dbl> <dbl>  ## 1 1 -0.594 1.02  ## 2 2 0.744 2.72  ## 3 3 0.931 3.93  ## 4 4 0.720 1.97  ## 5 5 -0.0281 1.95  ## 6 6 -0.0983 3.21  ## 7 7 0.126 0.926  ## 8 8 1.45 2.38  ## 9 9 0.298 1.66  ## 10 10 0.559 2.10 |

|  |  |
| --- | --- |
|  | Vous *pouvez* nommer chaque nouvel objet data et remplacer l’ancien objet data par le nouvel objet data à chaque étape. Cela permet de conserver un environnement propre. Cependant, je ne recommande pas cette option, car elle augmente les risques de confusion lorsque le code est exécuté ligne par ligne. |

Une manière d’éviter de créer des objets qui ne sont pas indispensables est de nicher les fonctions les unes dans les autres. Concrètement, cela consiste à remplacer chaque nouvel objet data par le code qui a permis de le générer à l’étape précédente. Cette option est élégante pour de courts enchaînements d’opérations.

|  |
| --- |
| mean\_petal\_width <- **round**(**mean**(iris$Petal.Width), 2) |

Mais elle porte rapidement à confusion pour des enchaînements plus longs:

|  |
| --- |
| # do not ever do this!!  data <- **rename**(  **spread**(  **summarise**(  **group\_by**(  **separate**(  **gather**(  **tibble**(  id = 1:10,  A1 = **rnorm**(10, 0),  A2 = **rnorm**(10, 1),  B1 = **rnorm**(10, 2),  B2 = **rnorm**(10,3)),  variable, value, A1:B2),  variable, c("var", "var\_n"), sep = 1),  id, var),  mean = **mean**(value)),  var, mean),  A\_mean = A, B\_mean = B) |

Le *pipe* permet “d’acheminer” le résultat de chaque fonction à la fonction suivante. Cela permet de réorganiser le code dans l’ordre logique (l’ordre dans lequel on pense aux opérations réalisées) sans créer de multiples objets.

|  |
| --- |
| # calculate mean of A and B variables for each participant  data <- **tibble**(  id = 1:10,  A1 = **rnorm**(10, 0),  A2 = **rnorm**(10, 1),  B1 = **rnorm**(10, 2),  B2 = **rnorm**(10,3)  ) %>%  **gather**(variable, value, A1:B2) %>%  **separate**(variable, **c**("var", "var\_n"), sep = 1) %>%  **group\_by**(id, var) %>%  **summarise**(mean = **mean**(value)) %>%  **spread**(var, mean) %>%  **rename**(A\_mean = A, B\_mean = B) |

Ce code peut se lire de haut en bas de la manière suivante:

1. Créer une *tibble* nommée data avec:
   * id allant de 1 à 10,
   * A1 étant composée de 10 nombres aléatoires issus d’une distribution normale,
   * A2 étant composée de 10 nombres aléatoires issus d’une distribution normale,
   * B1 étant composée de 10 nombres aléatoires issus d’une distribution normale,
   * B2 étant composée de 10 nombres aléatoires issus d’une distribution normale; et ensuite:
2. Rassembler (*gather*) les colonnes allant de A\_1 à B\_2 pour créer les colonnes variable et value; et ensuite
3. Séparer (*separate*) la colonne variable en deux nouvelles colonnes var et var\_n, séparer au caractère 1; et ensuite
4. Grouper (*group by*) par les colonnes id et var; et ensuite
5. Résumer (*summarise*) les données et créer une nouvelle colonne nommée mean qui contient la moyenne de la colonne value pour chaque groupe; et ensuite
6. Étaler (*spread*) les valeurs contenues dans la colonne mean en fonction des labels (key) contenus dans la colonne var; et ensuite
7. Renommer (*rename*) la colonne A en A\_mean et la colonne B en B\_mean.

Vous pouvez créer des objets intermédiaires afin de simplifier le code lorsque l’enchaînement devient trop compliqué (trop long), ou lorsque vous avez besoin de déboguer quelque chose.

|  |  |
| --- | --- |
|  | Vous pouvez déboguer un *pipe* en surlignant le code du début du *pipe* jusqu’à un point juste avant le *pipe* auquel vous souhaitez vous arrêter. Essayez cette astuce en surlignant allant de data <- jusqu'à la fin de la fonction separate et appuyez sur cmd-return. À quoi ressemble data désormais ? |

Nous allons enchaîner toutes les étapes ci-dessus en utilisant des *pipes*.

|  |
| --- |
| ocean <- **read\_csv**("https://psyteachr.github.io/msc-data-skills/data/personality.csv") %>%  **gather**("question", "score", Op1:Ex9) %>%  **separate**(question, c("domain", "qnumber"), sep = 2) %>%  **unite**("domain\_n", domain, qnumber, sep = "\_Q") %>%  **spread**(domain\_n, score) |

|  |
| --- |
| ## Parsed with column specification:  ## cols(  ## .default = col\_double(),  ## date = col\_date(format = "")  ## ) |

|  |
| --- |
| ## See spec(...) for full column specifications. |

# 4.7 Quelques exemples plus compliqués

## 4.7.1 Charger les données

Télécharger les données contenues dans le fichier CSV [infmort.csv](https://psyteachr.github.io/msc-data-skills/data/infmort.csv).

|  |
| --- |
| infmort <- **read\_csv**("data/infmort.csv") |

|  |
| --- |
| ## Parsed with column specification:  ## cols(  ## Country = col\_character(),  ## Year = col\_double(),  ## `Infant mortality rate (probability of dying between birth and age 1 per 1000 live births)` = col\_character()  ## ) |

|  |
| --- |
| **glimpse**(infmort) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 3  ## $ Country <chr> …  ## $ Year <dbl> …  ## $ `Infant mortality rate (probability of dying between birth and age 1 per 1000 live births)` <chr> … |

Télécharger les données sur la mortalité maternelle contenues dans le fichier Excel [.xls](https://psyteachr.github.io/msc-data-skills/data/matmort.xls).

|  |
| --- |
| matmort <- **read\_xls**("data/matmort.xls")  **glimpse**(matmort) |

|  |
| --- |
| ## Observations: 181  ## Variables: 4  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ `1990` <chr> "1 340 [ 878 - 1 950]", "71 [ 58 - 88]", "216 [ 141 - …  ## $ `2000` <chr> "1 100 [ 745 - 1 570]", "43 [ 33 - 56]", "170 [ 118 - …  ## $ `2015` <chr> "396 [ 253 - 620]", "29 [ 16 - 46]", "140 [ 82 - 244]… |

Télécharger les données sur les indicatifs de pays en cliquant sur le lien suivant: <https://raw.githubusercontent.com/lukes/ISO-3166-Countries-with-Regional-Codes/master/all/all.csv>

|  |
| --- |
| ccodes <- **read\_csv**("https://raw.githubusercontent.com/lukes/ISO-3166-Countries-with-Regional-Codes/master/all/all.csv") |

|  |
| --- |
| ## Parsed with column specification:  ## cols(  ## name = col\_character(),  ## `alpha-2` = col\_character(),  ## `alpha-3` = col\_character(),  ## `country-code` = col\_character(),  ## `iso\_3166-2` = col\_character(),  ## region = col\_character(),  ## `sub-region` = col\_character(),  ## `intermediate-region` = col\_character(),  ## `region-code` = col\_character(),  ## `sub-region-code` = col\_character(),  ## `intermediate-region-code` = col\_character()  ## ) |

|  |
| --- |
| **glimpse**(ccodes) |

|  |
| --- |
| ## Observations: 249  ## Variables: 11  ## $ name <chr> "Afghanistan", "Åland Islands", "Alba…  ## $ `alpha-2` <chr> "AF", "AX", "AL", "DZ", "AS", "AD", "…  ## $ `alpha-3` <chr> "AFG", "ALA", "ALB", "DZA", "ASM", "A…  ## $ `country-code` <chr> "004", "248", "008", "012", "016", "0…  ## $ `iso\_3166-2` <chr> "ISO 3166-2:AF", "ISO 3166-2:AX", "IS…  ## $ region <chr> "Asia", "Europe", "Europe", "Africa",…  ## $ `sub-region` <chr> "Southern Asia", "Northern Europe", "…  ## $ `intermediate-region` <chr> NA, NA, NA, NA, NA, NA, "Middle Afric…  ## $ `region-code` <chr> "142", "150", "150", "002", "009", "1…  ## $ `sub-region-code` <chr> "034", "154", "039", "015", "061", "0…  ## $ `intermediate-region-code` <chr> NA, NA, NA, NA, NA, NA, "017", "029",… |

## 4.7.2 Du format large au format long

matmort est au format large, avec une colonne par année. Transformez ce jeu de données au format long, avec une observation par ligne.

Cet exemple est un peu plus compliqués que les précédents car les noms de colonnes à regrouper sont des nombres. Lorsque les noms de colonnes ne sont pas standards (par exemple lorsqu’ils comprennent des espaces, qu’ils commencent avec des nombres ou lorsqu’ils contiennent des caractères spéciaux), il est possible d’entourer ces noms de colonnes en utilisant des *backticks* (`) comme dans l’exemple ci-dessous.

|  |
| --- |
| matmort\_long <- matmort %>%  **gather**("Year", "stats", `1990`:`2015`)  **glimpse**(matmort\_long) |

|  |
| --- |
| ## Observations: 543  ## Variables: 3  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ Year <chr> "1990", "1990", "1990", "1990", "1990", "1990", "1990", …  ## $ stats <chr> "1 340 [ 878 - 1 950]", "71 [ 58 - 88]", "216 [ 141 - … |

## 4.7.3 Une donnée par colonne

Les données dans la colonne stats sont dans un format inhabituel une sorte d’intervalle de confiance entre crochets et de nombreux espaces. Nous n’avons pas besoin de ces espaces et nous allons donc les supprimer en utilisant la fonction mutate.

La fonction separate function sert à séparer des données en utilisant n’importe quel caractères qui n’est pas ni une nombre ni une lettre (par défaut). Pour commencer, utilisez donc cette fonction sans spécifier l’argument sep. L’argument into spécifie une liste comprenant les noms des nouvelles colonnes.

|  |
| --- |
| matmort\_split <- matmort\_long %>%  mutate(stats = gsub(" ", "", stats)) %>%  separate(stats, c("rate", "ci\_low", "ci\_hi")) |

|  |
| --- |
| ## Warning: Expected 3 pieces. Additional pieces discarded in  ## 543 rows [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ## 16, 17, 18, 19, 20, ...]. |

|  |
| --- |
| glimpse(matmort\_split) |

|  |
| --- |
| ## Observations: 543  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ Year <chr> "1990", "1990", "1990", "1990", "1990", "1990", "1990", …  ## $ rate <chr> "1340", "71", "216", "1160", "72", "58", "8", "8", "64",…  ## $ ci\_low <chr> "878", "58", "141", "627", "64", "51", "7", "7", "56", "…  ## $ ci\_hi <chr> "1950", "88", "327", "2020", "80", "65", "9", "10", "74"… |

|  |  |
| --- | --- |
|  | La fonction gsub(pattern, replacement, x) offre une manière flexible de chercher et remplacer des éléments. L’exemple ci-dessus remplace toutes les occurrences du pattern " " (un espace), par le replacement "" (rien), dans la chaîne x (la colonne stats). Utilisez sub() au lieu de gsub() si vous souhaitez remplacez seulement la première occurrence d’un pattern donné. Nous avons utilisé un pattern relativement simple dans cet exemple, mais il est possible d’utiliser des patterns [regex](https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html) plus compliqués pour remplacer, par exemple, tous les nombres pairs (e.g., gsub(“[:02468:]”, "“,”id = 123456“)) ou toutes les occurrences du mot “colour” écrit en Anglais UK ou en Anglais US (e.g., gsub(”colo(u)?r“,”\*\*“,”replace color, colour, or colours, but not collors")). |

#### **4.7.3.1 Handle spare columns with extra**

|  |  |
| --- | --- |
|  | L’exemple précédent devrait vous retourner l’erreur suivante: “Too many values at 543 locations”. Cette erreur se produit car separate sépare la colonne à chaque crochet ou guillemet. Par conséquent, la chaîne de caractères 100[90-110] est divisée en 4 valeurs c(“100”, “90”, “110”, "“), mais nous avons seulement spécifié 3 noms de colonnes… Dans cet exemple, la quatrième valeur est toujours vide (la partie après le dernier crochet), donc ce comportement ne nous dérange pas, mais la fonction separate génère un avertissement afin que nous ne le fassions pas de manière accidentelle. Vous pouvez éviter cet avertissement en utilisant l’argument extra et en lui spécifiant “drop”. Regardez l'aide de ??tidyr::separate pour plus de détails. |

|  |
| --- |
| matmort\_split <- matmort\_long %>%  mutate(stats = gsub(" ", "", stats)) %>%  separate(stats, c("rate", "ci\_low", "ci\_hi"), extra = "drop")  glimpse(matmort\_split) |

|  |
| --- |
| ## Observations: 543  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ Year <chr> "1990", "1990", "1990", "1990", "1990", "1990", "1990", …  ## $ rate <chr> "1340", "71", "216", "1160", "72", "58", "8", "8", "64",…  ## $ ci\_low <chr> "878", "58", "141", "627", "64", "51", "7", "7", "56", "…  ## $ ci\_hi <chr> "1950", "88", "327", "2020", "80", "65", "9", "10", "74"… |

#### **4.7.3.2 Définir des délimiteurs avec l’argument sep**

Maintenance, faites la même chose avec le jeu de données infmort. Ce jeu de données est déjà au format long, donc vous n’avez pas besoin d’utiliser la fonction gather. La troisième colonne a un nom très long, et vous pouvez donc utiliser son numéro de colonne (3).

|  |
| --- |
| infmort\_split <- infmort %>%  separate(3, c("rate", "ci\_low", "ci\_hi"), extra = "drop")  glimpse(infmort\_split) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate <chr> "66", "68", "69", "71", "73", "75", "76", "78", "80", "8…  ## $ ci\_low <chr> "3", "1", "9", "7", "4", "1", "8", "6", "4", "3", "4", "…  ## $ ci\_hi <chr> "52", "55", "58", "61", "64", "66", "69", "71", "73", "7… |

*Attends, ça ne fonctionne pas du tout…!* Cela sépare la colonne à chaque espace, crochet, *et* point. Nous voulons seulement séparer la colonne aux espaces, crochets et tirets. Nous devons donc manuellement définir l’argument sep pour correspondre aux délimiteurs que nous souhaitons utiliser. Au fait, lorsqu’il y a plus que quelques arguments spécifiés pour une fonction, il est plus lisible d’écrire un argument par ligne.

|  |  |
| --- | --- |
|  | Vous pouvez utiliser des [expression régulières](https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html) pour séparer des colonnes complexes. Ici, nous aimerions séparer une colonne à chaque tirer et crochet. Vous pouvez séparer une colonne à partir d’une liste de délimiteurs en les écrivant entre parenthèse, séparés par un “|”. C’est un petit peu plus compliqué car les crochets ont un sens spécial dans les expressions régulières, alors vous devez “échapper” le crochet de gauche en utilisant deux backslashes “\\”. |

|  |
| --- |
| infmort\_split <- infmort %>%  separate(  col = 3,  into = c("rate", "ci\_low", "ci\_hi"),  extra = "drop",  sep = "(\\[|-|])"  )  glimpse(infmort\_split) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate <chr> "66.3 ", "68.1 ", "69.9 ", "71.7 ", "73.4 ", "75.1 ", "7…  ## $ ci\_low <chr> "52.7", "55.7", "58.7", "61.6", "64.4", "66.9", "69.0", …  ## $ ci\_hi <chr> "83.9", "83.6", "83.5", "83.7", "84.2", "85.1", "86.1", … |

#### **4.7.3.3 Corriger les types de données avec convert**

C’est mieux. Remarquez le sigle <chr> à côté de rate, ci\_low and ci\_hi dans l’encart précédent. Cela veut dire que ces colonnes contiennent des caractères (des mots), et non des nombres ou des entiers. Cela peut poser des problèmes si (par exemple) vous essayez de calculer la moyenne de ces colonnes (on ne peut pas calculer la moyenne d’un mot ou de plusieurs mots). Cependant, nous pouvons régler ce problème en utilisant l’argument convert and et en lui donnant la valeur TRUE.

|  |
| --- |
| infmort\_split <- infmort %>%  separate(3, c("rate", "ci\_low", "ci\_hi"), extra = "drop", sep = "(\\[|-|])", convert = TRUE)  glimpse(infmort\_split) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate <dbl> 66.3, 68.1, 69.9, 71.7, 73.4, 75.1, 76.8, 78.6, 80.4, 82…  ## $ ci\_low <dbl> 52.7, 55.7, 58.7, 61.6, 64.4, 66.9, 69.0, 71.2, 73.4, 75…  ## $ ci\_hi <dbl> 83.9, 83.6, 83.5, 83.7, 84.2, 85.1, 86.1, 87.3, 88.9, 90… |

Faites la même chose avec le jeu de données matmort.

|  |
| --- |
| matmort\_split <- matmort\_long %>%  mutate(stats = gsub(" ", "", stats)) %>%  separate(stats, c("rate", "ci\_low", "ci\_hi"), extra = "drop", convert = TRUE)  glimpse(matmort\_split) |

|  |
| --- |
| ## Observations: 543  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ Year <chr> "1990", "1990", "1990", "1990", "1990", "1990", "1990", …  ## $ rate <int> 1340, 71, 216, 1160, 72, 58, 8, 8, 64, 46, 26, 569, 58, …  ## $ ci\_low <int> 878, 58, 141, 627, 64, 51, 7, 7, 56, 34, 20, 446, 47, 28…  ## $ ci\_hi <int> 1950, 88, 327, 2020, 80, 65, 9, 10, 74, 61, 33, 715, 72,… |

### **4.7.4 Tout faire d’un coup**

Étant donné que nous n’avons pas besoin des jeux de données intermédiaires, nous pouvons enchainer toutes les étapes précédents.

|  |
| --- |
| infmort <- read\_csv("data/infmort.csv") %>%  separate(  3,  c("rate", "ci\_low", "ci\_hi"),  extra = "drop",  sep = "(\\[|-|])",  convert = TRUE  ) |

|  |
| --- |
| ## Parsed with column specification:  ## cols(  ## Country = col\_character(),  ## Year = col\_double(),  ## `Infant mortality rate (probability of dying between birth and age 1 per 1000 live births)` = col\_character()  ## ) |

|  |
| --- |
| matmort <- read\_xls("data/matmort.xls") %>%  gather("Year", "stats", `1990`:`2015`) %>%  mutate(stats = gsub(" ", "", stats)) %>%  separate(  stats,  c("rate", "ci\_low", "ci\_hi"),  extra = "drop",  convert = TRUE  )  glimpse(matmort)  glimpse(infmort) |

|  |
| --- |
| ## Observations: 543  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Angola", "Argentin…  ## $ Year <chr> "1990", "1990", "1990", "1990", "1990", "1990", "1990", …  ## $ rate <int> 1340, 71, 216, 1160, 72, 58, 8, 8, 64, 46, 26, 569, 58, …  ## $ ci\_low <int> 878, 58, 141, 627, 64, 51, 7, 7, 56, 34, 20, 446, 47, 28…  ## $ ci\_hi <int> 1950, 88, 327, 2020, 80, 65, 9, 10, 74, 61, 33, 715, 72,…  ## Observations: 5,044  ## Variables: 5  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate <dbl> 66.3, 68.1, 69.9, 71.7, 73.4, 75.1, 76.8, 78.6, 80.4, 82…  ## $ ci\_low <dbl> 52.7, 55.7, 58.7, 61.6, 64.4, 66.9, 69.0, 71.2, 73.4, 75…  ## $ ci\_hi <dbl> 83.9, 83.6, 83.5, 83.7, 84.2, 85.1, 86.1, 87.3, 88.9, 90… |

### **4.7.5 Colonnes par année**

Répartissez le taux de mortalité infantile par année.

|  |
| --- |
| infmort\_wide <- infmort %>%  spread(Year, rate)  glimpse(infmort\_wide) |

|  |
| --- |
| ## Observations: 4,934  ## Variables: 29  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ ci\_low <dbl> 52.7, 55.7, 58.7, 61.6, 64.4, 66.9, 69.0, 71.2, 73.4, 75…  ## $ ci\_hi <dbl> 83.9, 83.6, 83.5, 83.7, 84.2, 85.1, 86.1, 87.3, 88.9, 90…  ## $ `1990` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1991` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1992` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1993` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1994` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1995` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1996` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1997` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1998` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `1999` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `2000` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `2001` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, …  ## $ `2002` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 91.2…  ## $ `2003` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 89, NA, …  ## $ `2004` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 86.7, NA, NA…  ## $ `2005` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 84.4, NA, NA, NA…  ## $ `2006` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, 82.3, NA, NA, NA, NA…  ## $ `2007` <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 80.4, NA, NA, NA, NA, NA…  ## $ `2008` <dbl> NA, NA, NA, NA, NA, NA, NA, 78.6, NA, NA, NA, NA, NA, NA…  ## $ `2009` <dbl> NA, NA, NA, NA, NA, NA, 76.8, NA, NA, NA, NA, NA, NA, NA…  ## $ `2010` <dbl> NA, NA, NA, NA, NA, 75.1, NA, NA, NA, NA, NA, NA, NA, NA…  ## $ `2011` <dbl> NA, NA, NA, NA, 73.4, NA, NA, NA, NA, NA, NA, NA, NA, NA…  ## $ `2012` <dbl> NA, NA, NA, 71.7, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…  ## $ `2013` <dbl> NA, NA, 69.9, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…  ## $ `2014` <dbl> NA, 68.1, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…  ## $ `2015` <dbl> 66.3, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA… |

|  |  |
| --- | --- |
|  | Non… cela ne fonctionne pas du tout. Il s’agit d’une erreur commune lorsqu’on essaye de répartir des données. Cette erreur se présente car la fonction spread matches on all the remaining columns, so Afghanistan with ci\_low of 52.7 is treated as a different observation than Afghanistan with ci\_low of 55.7. On peut régler ce problème en fusionnant les colonnes rate, ci\_low et ci\_hi ensemble. |

### **4.7.6 Fusionner des colonnes**

Fusionnez les colonnes rate, ci\_low, et ci\_hi ensemble.

|  |
| --- |
| infmort\_united <- infmort %>%  unite(rate\_ci, rate, ci\_low, ci\_hi)  glimpse(infmort\_united) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 3  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate\_ci <chr> "66.3\_52.7\_83.9", "68.1\_55.7\_83.6", "69.9\_58.7\_83.5", "7… |

#### **4.7.6.1 Contrôle la séparation avec l’argument sep**

La fonction unite() sépare des noms de colonnes fusionnées avec un *underscore* par défaut. Utilisez l’argument sep si vous souhaitez modifier ce comportement.

|  |
| --- |
| infmort\_united <- infmort %>%  unite(rate\_ci, rate, ci\_low, ci\_hi, sep = ", ")  glimpse(infmort\_united) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 3  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate\_ci <chr> "66.3, 52.7, 83.9", "68.1, 55.7, 83.6", "69.9, 58.7, 83.… |

|  |  |
| --- | --- |
|  | Que faire si souhaitez remettre le jeu de données dans son format initial, c’est à dire au format “rate [ci\_low - ci\_hi]” ? Dans ce cas là, mutate et paste représentent un meilleur choix que unite, mais vous devez vous débarrassez des colonnes rate, ci\_low et ci\_hi en utilisant la fonction select. Vous en apprendrez plus à propos de ces fonctions dans le chapitre [Data Manipulation](https://psyteachr.github.io/msc-data-skills/04_dplyr.html). |

|  |
| --- |
| infmort\_united <- infmort %>%  mutate(rate\_ci = paste0(rate, " [", ci\_low, " - ", ci\_hi, "]"))  glimpse(infmort\_united) |

|  |
| --- |
| ## Observations: 5,044  ## Variables: 6  ## $ Country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanista…  ## $ Year <dbl> 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 20…  ## $ rate <dbl> 66.3, 68.1, 69.9, 71.7, 73.4, 75.1, 76.8, 78.6, 80.4, 82…  ## $ ci\_low <dbl> 52.7, 55.7, 58.7, 61.6, 64.4, 66.9, 69.0, 71.2, 73.4, 75…  ## $ ci\_hi <dbl> 83.9, 83.6, 83.5, 83.7, 84.2, 85.1, 86.1, 87.3, 88.9, 90…  ## $ rate\_ci <chr> "66.3 [52.7 - 83.9]", "68.1 [55.7 - 83.6]", "69.9 [58.7 … |

Maintenant, essayons de répartir de jeu de données à partir de Year à nouveau. Remarquez qu’ici nous unissons les colonnes rate:ci\_hi, au lieu de rate, ci\_low, ci\_hi. Les deux points (:) signifient que nous souhaitons sélecter toutes les colonnes entre la première et dernière colonne citée. Regardez l’aide des fonctions ??tidyr::unite et ??tidyr::select pour d’autres manières de sélectionner les colonnes.

|  |
| --- |
| infmort\_wide <- infmort %>%  unite(rate\_ci, rate:ci\_hi, sep = ", ") %>%  spread(Year, rate\_ci)  glimpse(infmort\_wide) |

|  |
| --- |
| ## Observations: 194  ## Variables: 27  ## $ Country <chr> "Afghanistan", "Albania", "Algeria", "Andorra", "Angola"…  ## $ `1990` <chr> "122.5, 111.6, 135.5", "35.1, 31.3, 39.2", "39.7, 37.1, …  ## $ `1991` <chr> "118.3, 108, 129.9", "33.7, 30.2, 37.6", "38.8, 36.1, 41…  ## $ `1992` <chr> "114.4, 104.6, 125.2", "32.5, 29.2, 36.2", "38.1, 35.4, …  ## $ `1993` <chr> "110.9, 101.4, 120.9", "31.4, 28.2, 34.9", "37.5, 34.9, …  ## $ `1994` <chr> "107.7, 98.6, 117.2", "30.3, 27.1, 33.8", "36.9, 34.6, 3…  ## $ `1995` <chr> "105, 96.2, 114.1", "29.1, 26, 32.7", "36.3, 34.2, 38.4"…  ## $ `1996` <chr> "102.7, 94.5, 111.3", "27.9, 24.8, 31.5", "35.7, 34, 37.…  ## $ `1997` <chr> "100.7, 92.9, 109.1", "26.8, 23.6, 30.4", "35.1, 33.8, 3…  ## $ `1998` <chr> "98.9, 91.4, 107.2", "25.5, 22.4, 29.2", "34.7, 33.7, 35…  ## $ `1999` <chr> "97.2, 89.9, 105.4", "24.4, 21.2, 28.1", "34.4, 33.5, 35…  ## $ `2000` <chr> "95.4, 88.2, 103.6", "23.2, 20, 27", "33.9, 33.2, 34.7",…  ## $ `2001` <chr> "93.4, 86.3, 101.6", "22.1, 18.8, 26", "33.3, 32.7, 34",…  ## $ `2002` <chr> "91.2, 84.3, 99.3", "21, 17.6, 25.1", "32.4, 31.8, 33", …  ## $ `2003` <chr> "89, 82.1, 97", "20, 16.5, 24.3", "31.3, 30.7, 31.9", "3…  ## $ `2004` <chr> "86.7, 79.9, 94.8", "19.1, 15.4, 23.8", "30.1, 29.5, 30.…  ## $ `2005` <chr> "84.4, 77.7, 92.6", "18.3, 14.2, 23.4", "28.8, 28.3, 29.…  ## $ `2006` <chr> "82.3, 75.5, 90.7", "17.4, 13.2, 23.1", "27.6, 27, 28.1"…  ## $ `2007` <chr> "80.4, 73.4, 88.9", "16.7, 12.1, 22.9", "26.4, 25.9, 26.…  ## $ `2008` <chr> "78.6, 71.2, 87.3", "16, 11.2, 22.7", "25.3, 24.8, 25.7"…  ## $ `2009` <chr> "76.8, 69, 86.1", "15.4, 10.5, 22.6", "24.3, 23.8, 24.7"…  ## $ `2010` <chr> "75.1, 66.9, 85.1", "14.8, 9.8, 22.4", "23.5, 23, 23.9",…  ## $ `2011` <chr> "73.4, 64.4, 84.2", "14.3, 9.1, 22.3", "22.8, 22.4, 23.3…  ## $ `2012` <chr> "71.7, 61.6, 83.7", "13.8, 8.5, 22.2", "22.4, 22, 22.9",…  ## $ `2013` <chr> "69.9, 58.7, 83.5", "13.3, 7.9, 22.1", "22.1, 21.7, 22.7…  ## $ `2014` <chr> "68.1, 55.7, 83.6", "12.9, 7.5, 22.1", "22, 21.3, 22.7",…  ## $ `2015` <chr> "66.3, 52.7, 83.9", "12.5, 7, 22.2", "21.9, 20.8, 23", "… |

# 4.8 Quiz

…

# 4.9 Exercises

…