Exploiter {janitor} pour nettoyer les données

Marie Vaugoyeau

12 November 2024

Table of contents

Le but de `{janitor}` est de proposer des fonctions simples qui permettent de nettoyer des données brutes qui sont bof côté structure :

- doublons de lignes
- des colonnes vides
- des noms de colonnes répétés, non homogène ou même absent

Associé à ces fonctions de nettoyage, il y a `tabyls` qui permet de visualiser les données sous la forme de tableaux résumés.

1 import des packages

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(janitor)
```

```
Attachement du package : 'janitor'

Les objets suivants sont masqués depuis 'package:stats':

    chisq.test, fisher.test
```

> ⚠ Attention aux conflits
>
> Le chargement des deux packages à masquer 4 fonctions du package `{stats}`.

2 import des données

Les données ont été créées pour l'occasion et sont disponible sur GitHub.

J'ai choisi de ne pas utilisé les données créées par Sam Firke afin de changer mais n'hésites pas à aller voir ses exemples !

```
data <- readxl::read_xlsx("data/donnees.xlsx")
```

3 Regardons un peu les données

```
glimpse(data)
```

```
Rows: 22
Columns: 8
$ `Prénom Patient.e` <chr> "Paula", "Pierre", "Antoine", "Adrien", "Alice", "S~
$ `Sexe / genre`     <chr> "F", "M", "M", "M", "F", "F", "M", "F", "M", "F", "~
$ date               <chr> "2024-01-01", "2024-01-16", "2024-01-31", "2024-02-~
$ `Album in\r\ng/dL` <dbl> 3.6, 3.9, 3.6, 3.9, 4.1, 3.8, 3.7, 3.7, 3.7, 3.4, 3~
$ `Fructose mg/dL`   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ `Glucose mg/dL`    <dbl> 92.85714, 89.14286, 89.88571, 85.42857, 96.57143, 8~
$ `Na (mmol/L)`      <dbl> 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 1~
$ `Globulin g/dL`    <dbl> 2.3, 2.0, 2.0, 2.0, 1.9, 2.0, 2.2, 2.1, 2.1, 2.6, 2~
```

```
summary(data)
```

```
 Prénom Patient.e   Sexe / genre           date            Album in\r\ng/dL
 Length:22          Length:22          Length:22          Min.   :3.400
 Class :character   Class :character   Class :character   1st Qu.:3.700
 Mode  :character   Mode  :character   Mode  :character   Median :3.800
                                                          Mean   :3.809
                                                          3rd Qu.:3.900
                                                          Max.   :4.200
 Fructose mg/dL Glucose mg/dL    Na (mmol/L)  Globulin g/dL
 Mode:logical   Min.   :81.71   Min.   :150   Min.   :1.900
 NA's:22        1st Qu.:84.87   1st Qu.:150   1st Qu.:2.000
                Median :86.91   Median :150   Median :2.100
                Mean   :87.39   Mean   :150   Mean   :2.109
                3rd Qu.:89.70   3rd Qu.:150   3rd Qu.:2.175
                Max.   :96.57   Max.   :150   Max.   :2.600
```

Comme tu le vois peut-être les noms des colonnes ne sont pas terribles (présence d'un saut de ligne `\r`, présence de parenthèses, majuscules…).
Il y a aussi des dates mal traduites dans la colonne `date` et une colonne vide

4 Améliorer les noms des colonnes

L'uniformisation des noms est facile grâce à la fonction `clean_names()`.

```
(donnees <- data |>
  clean_names())
```

```
# A tibble: 22 x 8
   prenom_patient_e sexe_genre date       album_in_g_d_l fructose_mg_d_l
   <chr>            <chr>      <chr>               <dbl> <lgl>
 1 Paula            F          2024-01-01            3.6 NA
 2 Pierre           M          2024-01-16            3.9 NA
 3 Antoine          M          2024-01-31            3.6 NA
 4 Adrien           M          2024-02-15            3.9 NA
 5 Alice            F          2024-03-01            4.1 NA
 6 Sarah            F          45398                 3.8 NA
 7 Louis            M          45393                 3.7 NA
 8 Sophie           F          45307                 3.7 NA
 9 Martin           M          2024-02-16            3.7 NA
10 Malmö            F          2024-03-02            3.4 NA
# i 12 more rows
```

```
# i 3 more variables: glucose_mg_d_l <dbl>, na_mmol_l <dbl>,
#   globulin_g_d_l <dbl>
```

Artwork by @allison_horst

Il est possible de choisir le format de colonne comme le `BigCamel`, `lowerCamel`, `SCREAMING_SNAKE` et d'autre.

```
(donnees <- readxl::read_xlsx("data/donnees.xlsx") |>
  clean_names(case = "lower_camel"))
```

```
# A tibble: 22 x 8
   prenomPatientE sexeGenre date     albumInGDL fructoseMgDL glucoseMgDL naMmolL
   <chr>          <chr>     <chr>         <dbl> <lgl>              <dbl>   <dbl>
 1 Paula          F         2024-01~        3.6 NA                  92.9     150
 2 Pierre         M         2024-01~        3.9 NA                  89.1     150
 3 Antoine        M         2024-01~        3.6 NA                  89.9     150
 4 Adrien         M         2024-02~        3.9 NA                  85.4     150
 5 Alice          F         2024-03~        4.1 NA                  96.6     150
 6 Sarah          F         45398           3.8 NA                  83.9     150
 7 Louis          M         45393           3.7 NA                  89.9     150
 8 Sophie         F         45307           3.7 NA                  83.9     150
 9 Martin         M         2024-02~        3.7 NA                  86.9     150
10 Malmö          F         2024-03~        3.4 NA                  83.2     150
# i 12 more rows
# i 1 more variable: globulinGDL <dbl>
```

Je vais conserver la forme par défaut qui est le `snake_case` que j'utilise d'habitude.
La fonction `make_clean_name()` permet la même chose mais sur un vecteur de nom. Ici, appliqué sur la colonne `prenom_patient_e` elle me permet d'uniformiser les prénoms en retirant les accents, les majuscules, les tirets…

```
(donnees <- readxl::read_xlsx("data/donnees.xlsx") |>
  clean_names() |>
  mutate(
    prenom_patient_e =
      prenom_patient_e |>
      make_clean_names()
  ))
```

```
# A tibble: 22 x 8
   prenom_patient_e sexe_genre date       album_in_g_d_l fructose_mg_d_l
   <chr>            <chr>      <chr>               <dbl> <lgl>
 1 paula            F          2024-01-01            3.6 NA
 2 pierre           M          2024-01-16            3.9 NA
 3 antoine          M          2024-01-31            3.6 NA
 4 adrien           M          2024-02-15            3.9 NA
 5 alice            F          2024-03-01            4.1 NA
 6 sarah            F          45398                 3.8 NA
 7 louis            M          45393                 3.7 NA
 8 sophie           F          45307                 3.7 NA
 9 martin           M          2024-02-16            3.7 NA
10 malmo            F          2024-03-02            3.4 NA
# i 12 more rows
# i 3 more variables: glucose_mg_d_l <dbl>, na_mmol_l <dbl>,
#   globulin_g_d_l <dbl>
```

5 retirer les colonnes vides

La fonction `remove_empty()` permet de retirer les lignes et/ou les colonnes vides.

```
(donnees <- readxl::read_xlsx("data/donnees.xlsx") |>
  clean_names() |>
  mutate(
    prenom_patient_e = make_clean_names(prenom_patient_e)
  ) |>
  remove_empty())
```

```
# A tibble: 22 x 7
   prenom_patient_e sexe_genre date      album_in_g_d_l glucose_mg_d_l na_mmol_l
   <chr>            <chr>      <chr>              <dbl>          <dbl>     <dbl>
 1 paula            F          2024-01-~            3.6           92.9       150
 2 pierre           M          2024-01-~            3.9           89.1       150
 3 antoine          M          2024-01-~            3.6           89.9       150
 4 adrien           M          2024-02-~            3.9           85.4       150
 5 alice            F          2024-03-~            4.1           96.6       150
 6 sarah            F          45398                3.8           83.9       150
 7 louis            M          45393                3.7           89.9       150
 8 sophie           F          45307                3.7           83.9       150
 9 martin           M          2024-02-~            3.7           86.9       150
10 malmo            F          2024-03-~            3.4           83.2       150
# i 12 more rows
# i 1 more variable: globulin_g_d_l <dbl>
```

> **Note**
>
> L'argument `which` par défaut est `c("rows", "cols")`. C'est cet argument qui permet de choisir de supprimer les lignes et/ou les colonnes vides.

Une fonction associée est `remove_constant()` qui retire les **colonnes constantes** (donc les colonnes vides aussi).

```
(donnees <- readxl::read_xlsx("data/donnees.xlsx") |>
  clean_names() |>
  mutate(
    prenom_patient_e =
      prenom_patient_e |>
      make_clean_names()
  ) |>
  remove_constant())
```

```
# A tibble: 22 x 6
   prenom_patient_e sexe_genre date       album_in_g_d_l glucose_mg_d_l
   <chr>            <chr>      <chr>               <dbl>          <dbl>
 1 paula            F          2024-01-01            3.6           92.9
 2 pierre           M          2024-01-16            3.9           89.1
 3 antoine          M          2024-01-31            3.6           89.9
 4 adrien           M          2024-02-15            3.9           85.4
 5 alice            F          2024-03-01            4.1           96.6
 6 sarah            F          45398                 3.8           83.9
```

```
 7 louis            M          45393                3.7           89.9
 8 sophie           F          45307                3.7           83.9
 9 martin           M          2024-02-16           3.7           86.9
10 malmo            F          2024-03-02           3.4           83.2
# i 12 more rows
# i 1 more variable: globulin_g_d_l <dbl>
```

6 traiter les dates excel

Lors du live, je t'ai montré les différentes étapes pour arriver au code ci-dessous qui permet de remplacer les dates format Excel (comme `45398` par exemple) sous un format date `yyyy-mm-dd`.

```
(donnees <- readxl::read_xlsx("data/donnees.xlsx") |>
  clean_names() |>
  mutate(
    prenom_patient_e =
      prenom_patient_e |>
      make_clean_names()
  ) |>
  remove_constant() |>
  mutate(
    date =
      case_when(
        str_detect(date, "-") ~ date,
        TRUE ~ date |>
          as.numeric() |>
          excel_numeric_to_date() |>
          as.character()
      ) |>
      ymd()
  ))
```

```
# A tibble: 22 x 6
   prenom_patient_e sexe_genre date       album_in_g_d_l glucose_mg_d_l
   <chr>            <chr>      <date>              <dbl>          <dbl>
 1 paula            F          2024-01-01            3.6           92.9
 2 pierre           M          2024-01-16            3.9           89.1
 3 antoine          M          2024-01-31            3.6           89.9
 4 adrien           M          2024-02-15            3.9           85.4
 5 alice            F          2024-03-01            4.1           96.6
```

```
 6 sarah            F          2024-04-16           3.8           83.9
 7 louis            M          2024-04-11           3.7           89.9
 8 sophie           F          2024-01-16           3.7           83.9
 9 martin           M          2024-02-16           3.7           86.9
10 malmo            F          2024-03-02           3.4           83.2
# i 12 more rows
# i 1 more variable: globulin_g_d_l <dbl>
```

7 créer des tableaux résumés rapidement

En plus de bien nettoyer les données, le package {janitor} permet de réaliser rapidement et simplement des tableaux résumés avec des analyses de données.

La première fonction à utiliser est `tabyl()`, une fonction un peu similaire à `count()` de {dplyr}.

```
tabyl(donnees, date)
```

```
      date n    percent
 2024-01-01 1 0.04545455
 2024-01-10 1 0.04545455
 2024-01-12 1 0.04545455
 2024-01-13 1 0.04545455
 2024-01-15 1 0.04545455
 2024-01-16 3 0.13636364
 2024-01-18 1 0.04545455
 2024-01-19 2 0.09090909
 2024-01-31 1 0.04545455
 2024-02-15 1 0.04545455
 2024-02-16 1 0.04545455
 2024-02-17 1 0.04545455
 2024-02-18 1 0.04545455
 2024-03-01 1 0.04545455
 2024-03-02 1 0.04545455
 2024-03-03 1 0.04545455
 2024-03-04 1 0.04545455
 2024-04-11 1 0.04545455
 2024-04-16 1 0.04545455
```

Au contraire de `count()`, `tabyl()` génère des tableaux croisés sous le même format que `table()` du package {base}.

```
tabyl(donnees, sexe_genre, date)
```

```
 sexe_genre 2024-01-01 2024-01-10 2024-01-12 2024-01-13 2024-01-15 2024-01-16
          F          1          0          0          0          1          2
          M          0          1          1          1          0          1
 2024-01-18 2024-01-19 2024-01-31 2024-02-15 2024-02-16 2024-02-17 2024-02-18
          1          2          0          0          0          0          1
          0          0          1          1          1          1          0
 2024-03-01 2024-03-02 2024-03-03 2024-03-04 2024-04-11 2024-04-16
          1          1          1          0          0          1
          0          0          0          1          1          0
```

Il est possible d'ajouter d'un titre aux colonnes grâce à la fonction `adorn_title()`. Toutes les fonctions commençant par `adorn_` permettent de modifier le tableau généré.

```
tabyl(donnees, sexe_genre, date) |>
  adorn_title()
```

```
                 date
 sexe_genre 2024-01-01 2024-01-10 2024-01-12 2024-01-13 2024-01-15 2024-01-16
          F          1          0          0          0          1          2
          M          0          1          1          1          0          1

 2024-01-18 2024-01-19 2024-01-31 2024-02-15 2024-02-16 2024-02-17 2024-02-18
          1          2          0          0          0          0          1
          0          0          1          1          1          1          0

 2024-03-01 2024-03-02 2024-03-03 2024-03-04 2024-04-11 2024-04-16
          1          1          1          0          0          1
          0          0          0          1          1          0
```

La fonction `adorn_totals()` permet par exemple d'ajouter une colonne total

```
tabyl(donnees, sexe_genre, date) |>
  adorn_title() |>
  adorn_totals(where = c("row", "col"))
```

```
Error in adorn_totals(adorn_title(tabyl(donnees, sexe_genre, date)), where = c("row", : at le
```

> **Attention**
>
> Il faut faire attention à l'ordre des lignes !

```
tabyl(donnees, sexe_genre, date) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_title()
```

```
                date
 sexe_genre 2024-01-01 2024-01-10 2024-01-12 2024-01-13 2024-01-15 2024-01-16
          F          1          0          0          0          1          2
          M          0          1          1          1          0          1
      Total          1          1          1          1          1          3

 2024-01-18 2024-01-19 2024-01-31 2024-02-15 2024-02-16 2024-02-17 2024-02-18
          1          2          0          0          0          0          1
          0          0          1          1          1          1          0
          1          2          1          1          1          1          1

 2024-03-01 2024-03-02 2024-03-03 2024-03-04 2024-04-11 2024-04-16 Total
          1          1          1          0          0          1    12
          0          0          0          1          1          0    10
          1          1          1          1          1          1    22
```

Enfin les fonctions `adorn_percentages()` et `adorn_pct_formatting()` affichent les pourcentages pourcentages pour la première et le nombre entre parenthèses pour la deuxième.

```
tabyl(donnees, date, sexe_genre) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_percentages() |>
  adorn_pct_formatting(digits = 1) |>
  adorn_ns() |>
  adorn_title()
```

```
             sexe_genre
       date           F           M       Total
 2024-01-01 100.0%  (1)   0.0%  (0) 100.0%  (1)
 2024-01-10   0.0%  (0) 100.0%  (1) 100.0%  (1)
 2024-01-12   0.0%  (0) 100.0%  (1) 100.0%  (1)
 2024-01-13   0.0%  (0) 100.0%  (1) 100.0%  (1)
 2024-01-15 100.0%  (1)   0.0%  (0) 100.0%  (1)
```

```
2024-01-16  66.7%  (2)  33.3%  (1) 100.0%  (3)
2024-01-18 100.0%  (1)   0.0%  (0) 100.0%  (1)
2024-01-19 100.0%  (2)   0.0%  (0) 100.0%  (2)
2024-01-31   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-02-15   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-02-16   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-02-17   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-02-18 100.0%  (1)   0.0%  (0) 100.0%  (1)
2024-03-01 100.0%  (1)   0.0%  (0) 100.0%  (1)
2024-03-02 100.0%  (1)   0.0%  (0) 100.0%  (1)
2024-03-03 100.0%  (1)   0.0%  (0) 100.0%  (1)
2024-03-04   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-04-11   0.0%  (0) 100.0%  (1) 100.0%  (1)
2024-04-16 100.0%  (1)   0.0%  (0) 100.0%  (1)
     Total  54.5% (12)  45.5% (10) 100.0                          (22)
```

8 En savoir un peu plus sur moi

Bonjour,

Je suis Marie Vaugoyeau et je suis disponible pour des **missions en freelance** d'**accompagnement à la formation** à R et à l'analyse de données et/ou en **programmation** (reprise de scripts, bonnes pratiques de codage, développement de package). Ayant un **bagage recherche en écologie**, j'ai accompagné plusieurs chercheuses en biologie dans leurs analyses de données mais je suis ouverte à d'autres domaines.

Vous pouvez retrouver mes offres ici.

En plus de mes missions de consulting je diffuse mes savoirs en R et analyse de données sur plusieurs plateformes :

- J'ai écrit un **livre** aux éditions ENI
- Tous les mois je fais un **live sur Twitch** pour parler d'un package de R, d'une analyse
- Je rédige une **newsletter** de manière irrégulière pour parler de mes **inspirations** et transmettre **des trucs et astuces sur R**. Pour s'y inscrire, c'est par là. J'ai aussi un **blog** sur lequel vous pourrez retrouver une version de cet article.

Pour en savoir encore un peu plus sur moi, il y a LinkedIn et pour retrouver tous ces liens et plus encore, c'est ici

N'hésitez pas à me contacter sur marie.vaugoyeau@gmail.com !

Bonne journée

Marie