Lire un export avec readr et readxl

Des spécifications de colonnes plutôt que des devinettes, problems() comme habitude et non comme sauvetage, et les pièges d'Excel — feuilles multiples, en-tête qui n'est pas la ligne un, cellules fusionnées et dates en numéro de série.

Pierre Robentz Cassion

Leçon 2 sur 8

R et le tidyverse pour les données de programme — Un projet qui se rouvre

Ce que couvre cette leçon

- readr devine, et vous dit ce qu'il a deviné
- La devinette porte sur les 1 000 premières lignes
- Déclarez les colonnes
- `mean()` renvoie NA, et c'est une qualité
- `na` s'applique au fichier, non à une colonne
- `problems()` est une habitude, non un sauvetage
- Vérifier la lecture avant d'en faire quoi que ce soit
- Séparateurs, encodages et milliers
- Excel
- La fonction de lecture, assemblée
- Ce qui vient ensuite

readr devine, et vous dit ce qu'il a deviné — En R

```r
library(readr)
library(here)

PATH <- here("data", "raw", "muac-screening-artibonite-2024.v1.csv")
muac <- read_csv(PATH)
```

readr devine, et vous dit ce qu'il a deviné — Exemple

```
child_id       character
commune        character
screening_date date
age_months     numeric
sex            character
muac_mm        numeric
oedema         character
outcome        character
```

La devinette porte sur les 1 000 premières lignes — En R

```r
# Deliberement : tout lire en texte, regarder, puis convertir.
peek <- read_csv(PATH, col_types = cols(.default = col_character()), n_max = 200)
print(peek)
```

Déclarez les colonnes — En R

```r
muac <- read_csv(
  PATH,
  col_types = cols(
    child_id       = col_character(),
    commune        = col_character(),
    screening_date = col_date(format = "%Y-%m-%d"),
    age_months     = col_integer(),
    sex            = col_character(),
    muac_mm        = col_integer(),
    oedema         = col_character(),
    outcome        = col_character()
  ),
  na = c("", "NA", "-99")
)
```

Déclarez les colonnes

- Les zéros initiaux survivent — `col_character()` sur un identifiant est toute la leçon : un code de formation sanitaire…
- Le format de date est énoncé — Un format inféré peut changer d'un fichier à l'autre selon la composition des valeurs
- La sentinelle est déclarée — Ce registre code un PB non mesuré par `-99`

Déclarez les colonnes — En R

```r
mean(muac$muac_mm)
#> [1] NA
```

`mean()` renvoie NA, et c'est une qualité — En R

```r
mean(muac$muac_mm)
#> [1] NA

mean(muac$muac_mm, na.rm = TRUE)
#> [1] 139.92
```

`mean()` renvoie NA, et c'est une qualité

- R est bruyant là où pandas est silencieux — Le `NA` est une question : *savez-vous qu'il y a ici des valeurs...

`mean()` renvoie NA, et c'est une qualité — En R

```r
sum(is.na(muac$muac_mm))
#> [1] 72
```

`na` s'applique au fichier, non à une colonne — En R

```r
muac <- read_csv(PATH, na = c("", "NA", "-99"))
```

`na` s'applique au fichier, non à une colonne — En R

```r
muac <- read_csv(PATH, na = c("", "NA")) |>
  dplyr::mutate(muac_mm = dplyr::na_if(muac_mm, -99L))
```

`problems()` est une habitude, non un sauvetage — En R

```
issues <- problems(muac)
nrow(issues)
#> [1] 0
```

`problems()` est une habitude, non un sauvetage — En R

```r
if (nrow(problems(muac)) > 0) {
  print(problems(muac))
  stop("L'entree ne correspond pas a sa specification de colonnes.")
}
```

Vérifier la lecture avant d'en faire quoi que ce soit — En R

```r
library(dplyr)

check <- function(df, expected_rows = NULL) {
  cat("lignes/colonnes ", nrow(df), ncol(df), "\n")
  print(dplyr::glimpse(df))
  print(colSums(is.na(df)))
  if (!is.null(expected_rows)) {
    stopifnot(nrow(df) == expected_rows)
  }
}

check(muac, expected_rows = 4218)
```

Séparateurs, encodages et milliers — En R

```r
epi <- read_csv2(PATH)                      # separateur ; decimale ,
epi <- read_delim(PATH, delim = "\t")       # tabulations
```

Séparateurs, encodages et milliers — En R

```r
muac <- read_csv(PATH, locale = locale(encoding = "UTF-8"))
muac <- read_csv(PATH, locale = locale(encoding = "latin1"))    # ce qu'Excel ecrit souvent
```

Séparateurs, encodages et milliers — En R

```r
epi <- read_csv(PATH, locale = locale(grouping_mark = ","))
```

Excel — En R

```r
library(readxl)

excel_sheets(path)
#> [1] "Cover" "Data" "Codebook" "Sheet3"

data <- read_excel(path, sheet = "Data")
```

Excel

- L'en-tête n'est pas la ligne un

Excel — En R

```r
data <- read_excel(path, sheet = "Data", skip = 3)
```

Excel

- Les cellules fusionnées

Excel — En R

```r
data <- data |> tidyr::fill(district, .direction = "down")
```

Excel

- Les dates en numéro de série

Excel — En R

```r
data$screening_date <- as.Date(as.numeric(data$screening_date), origin = "1899-12-30")
```

Excel

- Les types sous Excel

Excel — En R

```r
data <- read_excel(
  path,
  sheet = "Data",
  col_types = c("text", "text", "date", "numeric", "text", "numeric", "text", "text")
)
```

La fonction de lecture, assemblée — En R (suite)

```r
# R/read_register.R
read_register <- function(path) {
  muac <- readr::read_csv(
    path,
    col_types = readr::cols(
      child_id       = readr::col_character(),
      commune        = readr::col_character(),
      screening_date = readr::col_date(format = "%Y-%m-%d"),
      age_months     = readr::col_integer(),
      sex            = readr::col_character(),
      muac_mm        = readr::col_integer(),
      oedema         = readr::col_character(),
      outcome        = readr::col_character()
    ),
    na = c("", "NA", "-99")
  )
```

La fonction de lecture, assemblée — En R (suite)

```
  if (nrow(readr::problems(muac)) > 0) {
    print(readr::problems(muac))
    stop("L'entree ne correspond pas a sa specification de colonnes.")
  }
  stopifnot(!any(is.na(muac$child_id)))

  muac
}
```

Ce qui vient ensuite

- Le CSV est entré et ses types sont justes.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/r-for-programme-data/r-02-reading-exports