Lire un export sans le corrompre

Inférence de types, zéros initiaux, dates, et le code de valeur manquante qui devient un nombre. Les dégâts se produisent à l'import, avant même que vous ayez regardé quoi que ce soit.

Pierre Robentz Cassion

Leçon 3 sur 8

Fondamentaux de l'analyse de données pour le suivi-évaluation

Ce que couvre cette leçon

- Les dégâts précèdent le regard
- Les codes de valeur manquante deviennent des nombres
- Les identifiants ne sont pas des nombres
- Les dates
- Les catégories à vocabulaire fixe
- Une lecture défensive, de bout en bout
- Lire les autres formats
- La suite

Les dégâts précèdent le regard

- `read_csv(chemin)` tient sur une ligne et prend une demi-douzaine de décisions à votre place.

Les codes de valeur manquante deviennent des nombres — En Python

```
import pandas as pd

naif = pd.read_csv("data/raw/muac-screening-artibonite-2024.v1.csv")
print(naif["muac_mm"].mean())
```

Les codes de valeur manquante deviennent des nombres — En Python

```
muac = pd.read_csv(
    "data/raw/muac-screening-artibonite-2024.v1.csv",
    na_values={"muac_mm": ["-99"]},
)

print(muac["muac_mm"].mean())
print(muac["muac_mm"].isna().sum())
```

Les codes de valeur manquante deviennent des nombres — En R

```r
library(readr)

muac <- read_csv(
  "data/raw/muac-screening-artibonite-2024.v1.csv",
  na = c("", "NA", "-99")
)

mean(muac$muac_mm, na.rm = TRUE)
sum(is.na(muac$muac_mm))
```

Les codes de valeur manquante deviennent des nombres

Traiter une sentinelle n'est pas la même chose que décider quoi faire de la valeur manquante. Cette décision relève de la leçon 6. Ici, vous vous assurez seulement que le code cesse de se faire passer pour une mesure.

Les identifiants ne sont pas des nombres — En Python

```
muac = pd.read_csv(
    chemin,
    dtype={"child_id": "string", "commune": "string"},
    na_values={"muac_mm": ["-99"]},
)
```

Les identifiants ne sont pas des nombres — En R

```r
muac <- read_csv(
  chemin,
  col_types = cols(
    child_id = col_character(),
    commune  = col_character()
  ),
  na = c("", "NA", "-99")
)
```

Les dates — En Python

```
muac["screening_date"] = pd.to_datetime(
    muac["screening_date"], format="%Y-%m-%d", errors="raise"
)
```

Les dates — En R

```
muac <- muac |>
  dplyr::mutate(
    screening_date = as.Date(screening_date, format = "%Y-%m-%d")
  )

stopifnot(!any(is.na(muac$screening_date)))
```

Les dates

- **Énoncez explicitement le format** plutôt que de laisser l'analyseur le déduire. Un format déduit peut changer d'un…
- **Utilisez** `errors="raise"`**.** L'alternative, `errors="coerce"`, convertit toute date illisible en valeur manquante —…

Les catégories à vocabulaire fixe — En Python

```
issues = pd.CategoricalDtype(
    ["no-action", "referred-tsfp", "referred-otp", "referred-sc"], ordered=False
)

muac["outcome"] = muac["outcome"].astype(issues)

# Toute valeur hors vocabulaire est désormais NaN -- comptez-les avant de continuer.
print(muac["outcome"].isna().sum())
```

Les catégories à vocabulaire fixe — En R

```r
muac <- muac |>
  dplyr::mutate(
    outcome = factor(
      outcome,
      levels = c("no-action", "referred-tsfp", "referred-otp", "referred-sc")
    )
  )

sum(is.na(muac$outcome))
```

Une lecture défensive, de bout en bout — En Python (suite)

```
import pandas as pd

CHEMIN = "data/raw/muac-screening-artibonite-2024.v1.csv"

muac = pd.read_csv(
    CHEMIN,
    dtype={"child_id": "string", "commune": "string", "sex": "string"},
    na_values={"muac_mm": ["-99"]},
    keep_default_na=True,
)

muac["screening_date"] = pd.to_datetime(
    muac["screening_date"], format="%Y-%m-%d", errors="raise"
)

assert len(muac) == 4218, f"4218 lignes attendues, {len(muac)} obtenues"
```

Une lecture défensive, de bout en bout — En Python (suite)

```
assert muac["child_id"].notna().all(), "chaque ligne exige un identifiant"

print(muac.dtypes)
print(muac.isna().sum())
```

Une lecture défensive, de bout en bout — En R (suite)

```r
library(readr)
library(dplyr)

CHEMIN <- "data/raw/muac-screening-artibonite-2024.v1.csv"

muac <- read_csv(
  CHEMIN,
  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()
```

Une lecture défensive, de bout en bout — En R (suite)

```
  ),
  na = c("", "NA", "-99")
)

stopifnot(nrow(muac) == 4218)
stopifnot(!any(is.na(muac$child_id)))

glimpse(muac)
colSums(is.na(muac))
```

Lire les autres formats

Source	Python	R
CSV	`pd.read_csv`	`readr::read_csv`
Excel	`pd.read_excel`	`readxl::read_excel`
Stata	`pd.read_stata`	`haven::read_dta`
SPSS	`pd.read_spss`	`haven::read_sav`
Parquet	`pd.read_parquet`	`arrow::read_parquet`

La suite

- Le registre est maintenant en mémoire, avec ses types intacts.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/data-analysis-foundations/foundations-03-reading-exports