Obtenir un tableau ordonné

Restructurer les groupes répétés aplatis en une ligne par observation, joindre une liste de membres à son ménage, et prouver que la jointure n'a pas modifié votre charge de cas en silence.

Pierre Robentz Cassion

Leçon 4 sur 8

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

Ce que couvre cette leçon

- La règle
- À quoi ressemble le désordre dans ce secteur
- Format long et format large
- Joindre une liste à son parent
- Prouvez que la jointure a fait ce que vous aviez annoncé
- Le registre est déjà ordonné
- La suite

La règle

- Une ligne par observation, une colonne par variable, une table par type d'objet.

À quoi ressemble le désordre dans ce secteur

- Groupes répétés aplatis — CommCare exporte une ligne par soumission, donc une visite de ménage ayant dépisté trois…

À quoi ressemble le désordre dans ce secteur — Exemple

```
form_id,commune,child_1_id,child_1_muac,child_2_id,child_2_muac,child_3_id,child_3_muac
F0012,Gonaives,CH00854,133,CH00855,118,,
F0013,Verrettes,CH01439,143,,,,
```

À quoi ressemble le désordre dans ce secteur

- Des valeurs dans les en-têtes de colonnes — Un tableau croisé DHIS2 dispose les périodes en haut :

À quoi ressemble le désordre dans ce secteur — Exemple

```
org_unit,janv-2024,févr-2024,mars-2024
Gonaives,412,388,401
```

À quoi ressemble le désordre dans ce secteur

- Plusieurs objets dans une même table — Un export d'enquête où les caractéristiques du ménage sont répétées sur chaque…

Format long et format large — En Python (suite)

```python
import pandas as pd

large = pd.read_csv("data/raw/commcare-screening-export.csv")

long = large.melt(
    id_vars=["form_id", "commune"],
    var_name="champ",
    value_name="valeur",
)

# child_1_muac -> enfant 1, champ muac
parties = long["champ"].str.extract(r"^child_(?P<rang>\d+)_(?P<attribut>.+)$")
long = pd.concat([long, parties], axis=1).dropna(subset=["rang"])

enfants = (
    long.pivot(index=["form_id", "commune", "rang"], columns="attribut", values="valeur")
```

Format long et format large — En Python (suite)

```
    .reset_index()
    .dropna(subset=["id"])
)

print(enfants.head())
```

Format long et format large — En R (suite)

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

large <- read_csv("data/raw/commcare-screening-export.csv")

enfants <- large |>
  pivot_longer(
    cols = starts_with("child_"),
    names_to = c("rang", "attribut"),
    names_pattern = "child_(\\d+)_(.+)",
    values_to = "valeur",
    values_transform = as.character
  ) |>
  pivot_wider(names_from = attribut, values_from = valeur) |>
  filter(!is.na(id))
```

Format long et format large — En R (suite)

```
enfants
```

Format long et format large

- **Repérez le rang par une expression régulière**, jamais en énumérant les colonnes. La liste change d'un export à…
- **Supprimez les rangs vides.** Un formulaire ayant dépisté un seul enfant produit malgré tout des colonnes pour les…

Joindre une liste à son parent — En Python

```python
menages = pd.read_excel("data/raw/kobo-export.xlsx", sheet_name="household")
membres = pd.read_excel("data/raw/kobo-export.xlsx", sheet_name="member")

fusion = membres.merge(
    menages[["_index", "commune", "interview_date"]],
    left_on="_parent_index",
    right_on="_index",
    how="left",
    validate="many_to_one",
    indicator=True,
)

print(fusion["_merge"].value_counts())
```

Joindre une liste à son parent — En R

```r
library(readxl)

menages <- read_excel("data/raw/kobo-export.xlsx", sheet = "household")
membres <- read_excel("data/raw/kobo-export.xlsx", sheet = "member")

fusion <- membres |>
  left_join(
    menages |> select(`_index`, commune, interview_date),
    by = join_by(`_parent_index` == `_index`),
    relationship = "many-to-one",
    unmatched = "error"
  )

nrow(fusion) == nrow(membres)
```

Prouvez que la jointure a fait ce que vous aviez annoncé — En Python

```
avant = len(membres)
apres = len(fusion)

assert apres == avant, f"la jointure a changé le nombre de lignes : {avant} -> {apres}"

orphelins = (fusion["_merge"] == "left_only").sum()
assert orphelins == 0, f"{orphelins} membres sans ménage"

assert fusion["child_id"].is_unique, "identifiants dupliqués après jointure"
```

Prouvez que la jointure a fait ce que vous aviez annoncé — En R

```
stopifnot(nrow(fusion) == nrow(membres))
stopifnot(!any(is.na(fusion$commune)))
stopifnot(!any(duplicated(fusion$child_id)))
```

Prouvez que la jointure a fait ce que vous aviez annoncé

Une jointure qui modifie votre nombre de lignes n'est pas un problème de données à corriger en aval. C'est le signe que vous avez mal compris l'une des deux tables, et la correction se situe en amont de la jointure.

Le registre est déjà ordonné — En Python

```
muac.head()
```

Le registre est déjà ordonné — En R

```
head(muac)
```

La suite

- Le tableau a la bonne forme.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/data-analysis-foundations/foundations-04-tidy-tables