La même personne, deux fois, sous deux identifiants

L'appariement d'enregistrements quand rien ne joint — blocage, normalisation, distance d'édition et score. Douze paires candidates dans le registre de dépistage, six réelles, et quatre fabriquées par les âges manquants de la leçon 2.

Pierre Robentz Cassion

Leçon 4 sur 8

Nettoyage et validation des données — Identité et doublons

Ce que couvre cette leçon

- Le doublon sans clé
- Commencez par les attributs qui ne devraient pas coïncider
- Regardez les faux avant les vrais
- Le blocage, et pourquoi on ne peut pas s'en passer
- Normalisez avant de comparer
- Distance d'édition et choix d'un seuil
- Notez plusieurs champs, n'enchaînez pas les filtres
- Le produit est une file de relecture, pas un tableau propre
- Ce qu'il ne faut pas automatiser
- La suite

Le doublon sans clé

- Un enfant dépisté le matin, renvoyé chez lui, puis ramené l'après-midi par un autre accompagnant est enregistré deux fois sous deux identifiants.

Commencez par les attributs qui ne devraient pas coïncider — En Python

```python
CANDIDATE_KEY = ["commune", "screening_date", "age_months", "sex", "muac_mm"]

candidates = (
    muac.groupby(CANDIDATE_KEY, dropna=False)
    .filter(lambda g: g["child_id"].nunique() > 1)
    .sort_values(CANDIDATE_KEY)
)
print(candidates[["child_id"] + CANDIDATE_KEY].to_string(index=False))
```

Commencez par les attributs qui ne devraient pas coïncider — En R

```r
CANDIDATE_KEY <- c("commune", "screening_date", "age_months", "sex", "muac_mm")

candidates <- muac |>
  group_by(across(all_of(CANDIDATE_KEY))) |>
  filter(n_distinct(child_id) > 1) |>
  ungroup() |>
  arrange(across(all_of(CANDIDATE_KEY)))
```

Regardez les faux avant les vrais

Commune	Date	Âge	Sexe	PB	Identifiants
Desdunes	2024-01-20	36	f	134	CH03315, CH09241
Gros-Morne	2024-06-10	*manquant*	f	131	CH00576, CH02413
Gros-Morne	2024-06-10	*manquant*	m	123	CH01302, CH04050
Gros-Morne	2024-06-13	*manquant*	m	-99	CH00519, CH00755
Gros-Morne	2024-06-14	*manquant*	m	137	CH01558, CH02913
Verrettes	2024-11-09	26	f	127	CH02193, CH03129

Regardez les faux avant les vrais

- Une colonne de blocage comportant des valeurs manquantes fabrique des appariements — Excluez les lignes manquantes du...

Regardez les faux avant les vrais — En Python

```
blocked = muac[muac["age_months"].notna() & muac["muac_mm"].notna()]
```

Regardez les faux avant les vrais — En R

```
blocked <- muac |> filter(!is.na(age_months), !is.na(muac_mm))
```

Le blocage, et pourquoi on ne peut pas s'en passer — En Python

```
blocks = blocked.groupby(["commune", "sex"])
print(sum(len(g) * (len(g) - 1) // 2 for _, g in blocks), "pairs to compare")
```

Le blocage, et pourquoi on ne peut pas s'en passer — En R

```
blocked |>
  count(commune, sex) |>
  summarise(pairs = sum(n * (n - 1) / 2))
```

Normalisez avant de comparer — En Python (suite)

```
import re
import unicodedata


def normalise(series):
    return (
        series.fillna("")
        .str.normalize("NFKD")
        .str.encode("ascii", "ignore").str.decode("ascii")
        .str.lower()
        .str.replace(r"[^a-z0-9 ]", " ", regex=True)
        .str.replace(r"\s+", " ", regex=True)
        .str.strip()
    )
```

Normalisez avant de comparer — En Python (suite)

```
households["head_key"] = normalise(households["head_of_household"])
```

Normalisez avant de comparer — En R

```r
normalise <- function(x) {
  x |>
    tidyr::replace_na("") |>
    stringi::stri_trans_general("Latin-ASCII") |>
    tolower() |>
    stringr::str_replace_all("[^a-z0-9 ]", " ") |>
    stringr::str_squish()
}

households$head_key <- normalise(households$head_of_household)
```

Normalisez avant de comparer

- Ne retirez les accents que pour la clé de comparaison, jamais dans les données conservées — `Étienne` est le nom de la…

Distance d'édition et choix d'un seuil — En Python

```
from rapidfuzz import fuzz, process

matches = process.extract(
    "jean baptiste pierre",
    households["head_key"].tolist(),
    scorer=fuzz.token_sort_ratio,
    score_cutoff=88,
    limit=10,
)
```

Distance d'édition et choix d'un seuil — En R

```r
scores <- stringdist::stringsim(
  "jean baptiste pierre",
  households$head_key,
  method = "jw"
)
which(scores > 0.88)
```

Notez plusieurs champs, n'enchaînez pas les filtres — En Python

```
def pair_score(a, b):
    name = fuzz.token_sort_ratio(a["head_key"], b["head_key"]) / 100
    same_site = 1.0 if a["community"] == b["community"] else 0.0
    size_gap = abs(a["household_size"] - b["household_size"])
    size = max(0.0, 1 - size_gap / 5)
    return 0.6 * name + 0.25 * same_site + 0.15 * size
```

Notez plusieurs champs, n'enchaînez pas les filtres — En R

```r
pair_score <- function(a, b) {
  name      <- stringdist::stringsim(a$head_key, b$head_key, method = "jw")
  same_site <- as.numeric(a$community == b$community)
  size      <- pmax(0, 1 - abs(a$household_size - b$household_size) / 5)
  0.6 * name + 0.25 * same_site + 0.15 * size
}
```

Le produit est une file de relecture, pas un tableau propre — En Python

```
review = (
    pairs.sort_values("score", ascending=False)
    .loc[:, ["id_a", "id_b", "score", "head_a", "head_b", "community", "size_gap"]]
    .assign(decision="", reviewer="", reviewed_on="")
)
review.to_csv("outputs/review/duplicate-candidates.csv", index=False)
```

Le produit est une file de relecture, pas un tableau propre — En R

```r
review <- pairs |>
  arrange(desc(score)) |>
  select(id_a, id_b, score, head_a, head_b, community, size_gap) |>
  mutate(decision = "", reviewer = "", reviewed_on = "")

readr::write_csv(review, here::here("outputs", "review", "duplicate-candidates.csv"))
```

Le produit est une file de relecture, pas un tableau propre

Un doublon fusionné est un enregistrement détruit. Si la fusion était fausse, la preuve qu'elle l'était a disparu avec lui.

Ce qu'il ne faut pas automatiser

- **Une fausse fusion supprime quelqu'un.** Deux ménages n'en font plus qu'un ; l'un cesse de recevoir l'assistance sans…
- **Une fausse scission compte double.** C'est coûteux et embarrassant, et cela ne prive personne de rien.

La suite

- Vous savez désormais quelles lignes désignent la même chose et quelles colonnes les identifient.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/data-cleaning-and-validation/cleaning-04-duplicates-without-a-key