The same person, twice, under two identifiers

Record linkage when nothing joins — blocking, normalisation, edit distance and a score. Twelve candidate pairs in the screening register, six of them real, and four manufactured by the missing ages from lesson 2.

Pierre Robentz Cassion

Lesson 4 of 8

Data Cleaning and Validation — Identity and duplication

What this lesson covers

- The duplicate with no key
- Start with the attributes that should not coincide
- Look at the false ones before the real ones
- Blocking, and why you cannot skip it
- Normalise before you compare
- Edit distance, and choosing a threshold
- Score several fields, do not chain filters
- The output is a review queue, not a clean table
- What you must not automate
- What comes next

The duplicate with no key

- A child screened in the morning, sent home, and brought back in the afternoon by a different carer is registered twice with two identifiers.

Start with the attributes that should not coincide — In 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))
```

Start with the attributes that should not coincide — In 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)))
```

Look at the false ones before the real ones

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

Look at the false ones before the real ones

- A blocking column with missing values manufactures matches — Either exclude missing rows from the block or block on...

Look at the false ones before the real ones — In Python

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

Look at the false ones before the real ones — In R

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

Blocking, and why you cannot skip it

- Blocking — means only comparing records that already agree on something cheap and reliable — the commune, the...

Blocking, and why you cannot skip it — In Python

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

Blocking, and why you cannot skip it — In R

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

Normalise before you compare — In Python (cont.)

```
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()
    )
```

Normalise before you compare — In Python (cont.)

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

Normalise before you compare — In 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)
```

Normalise before you compare

- Strip accents for the comparison key only, never in the data you keep — `Étienne` is the person's name

Edit distance, and choosing a threshold — In 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,
)
```

Edit distance, and choosing a threshold — In R

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

Score several fields, do not chain filters — In 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
```

Score several fields, do not chain filters — In 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
}
```

The output is a review queue, not a clean table — In 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)
```

The output is a review queue, not a clean table — In 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"))
```

The output is a review queue, not a clean table

A duplicate you merged is a record you destroyed. If the merge was wrong, the evidence that it was wrong went with it.

What you must not automate

- **A false merge removes someone.** Two households become one; one of them stops receiving assistance and has no way to...
- **A false split double-counts.** Costly and embarrassing, and it does not deprive anyone of anything.

What comes next

- You now know which rows are the same thing and which columns identify them.

Read the full lesson, with runnable code

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