Getting to a tidy table

Reshape flattened repeat groups into one row per observation, join a member roster to its household, and prove the join did not silently change your caseload.

Pierre Robentz Cassion

Lesson 4 of 8

Data Analysis Foundations for M&E

What this lesson covers

- The rule
- What untidy looks like in this sector
- Long and wide
- Joining a roster to its parent
- Prove the join did what you said
- The register is already tidy
- What comes next

The rule

- One row per observation, one column per variable, one table per kind of thing.

What untidy looks like in this sector

- Flattened repeat groups — CommCare exports one row per form submission, so a household visit that screened three...

What untidy looks like in this sector — Example

```
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,,,,
```

What untidy looks like in this sector

- Values in column headers — A DHIS2 pivot puts periods across the top:

What untidy looks like in this sector — Example

```
org_unit,Jan-2024,Feb-2024,Mar-2024
Gonaives,412,388,401
```

What untidy looks like in this sector

- Multiple things in one table — A survey export with household characteristics repeated on every member row

Long and wide — In Python (cont.)

```python
import pandas as pd

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

long = wide.melt(
    id_vars=["form_id", "commune"],
    var_name="field",
    value_name="value",
)

# child_1_muac -> child 1, field muac
parts = long["field"].str.extract(r"^child_(?P<slot>\d+)_(?P<attribute>.+)$")
long = pd.concat([long, parts], axis=1).dropna(subset=["slot"])

children = (
    long.pivot(index=["form_id", "commune", "slot"], columns="attribute", values="value")
```

Long and wide — In Python (cont.)

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

print(children.head())
```

Long and wide — In R (cont.)

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

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

children <- wide |>
  pivot_longer(
    cols = starts_with("child_"),
    names_to = c("slot", "attribute"),
    names_pattern = "child_(\\d+)_(.+)",
    values_to = "value",
    values_transform = as.character
  ) |>
  pivot_wider(names_from = attribute, values_from = value) |>
  filter(!is.na(id))
```

Long and wide — In R (cont.)

```
children
```

Long and wide

- **Match the slot number with a pattern**, never by listing columns. The list changes between exports; the pattern does. . .
- **Drop the empty slots.** A form that screened one child still produces columns for slots two and three, filled with. . .

Joining a roster to its parent — In Python

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

merged = members.merge(
    households[["_index", "commune", "interview_date"]],
    left_on="_parent_index",
    right_on="_index",
    how="left",
    validate="many_to_one",
    indicator=True,
)

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

Joining a roster to its parent — In R

```r
library(readxl)

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

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

nrow(merged) == nrow(members)
```

Prove the join did what you said — In Python

```
before = len(members)
after = len(merged)

assert after == before, f"join changed the row count: {before} -> {after}"

unmatched = (merged["_merge"] == "left_only").sum()
assert unmatched == 0, f"{unmatched} members have no household"

assert merged["child_id"].is_unique, "duplicate identifiers after join"
```

Prove the join did what you said — In R

```
stopifnot(nrow(merged) == nrow(members))
stopifnot(!any(is.na(merged$commune)))
stopifnot(!any(duplicated(merged$child_id)))
```

Prove the join did what you said

A join that changes your row count is not a data problem you fix downstream. It is a signal that you misunderstood one of the two tables, and the fix is upstream of the join.

The register is already tidy — In Python

```
muac.head()
```

The register is already tidy — In R

```
head(muac)
```

What comes next

- The table is the right shape.

Read the full lesson, with runnable code

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