Long, wide, and the repeat group in between

Pivot because the analysis needs the shape, not because the export arrived in it. The flattened repeat group, the DHIS2 pivot table, and the fill value that turns "not reported" into zero.

Pierre Robentz Cassion

Lesson 4 of 8

Joining and Reshaping Programme Data — One row per what?

What this lesson covers

- Two shapes, and what each is for
- The flattened repeat group
- The empty slots, and the ones that mean something
- Long to wide: the DHIS2 pivot
- The fill value that invents data
- The wide table you cannot filter
- Round-trip, and check it
- What comes next

Two shapes, and what each is for

- Long — one row per observation, one column per variable
- Wide — one row per unit, one column per measurement

Two shapes, and what each is for

Long is for computing. Wide is for reading.

The flattened repeat group — Example

```
household_id, member_1_age, member_1_sex, member_2_age, member_2_sex, member_3_age, ...
```

The flattened repeat group — In Python

```
long = members_wide.melt(
    id_vars="household_id",
    var_name="field",
    value_name="value",
)
long[["slot", "attribute"]] = long["field"].str.extract(r"member_(\d+)_(\w+)")

members = (
    long.dropna(subset=["slot"])
    .pivot(index=["household_id", "slot"], columns="attribute", values="value")
    .reset_index()
)
```

The flattened repeat group — In R

```r
members <- members_wide |>
  tidyr::pivot_longer(
    cols = tidyr::starts_with("member_"),
    names_to = c("slot", "attribute"),
    names_pattern = "member_(\\d+)_(\\w+)",
    values_to = "value"
  ) |>
  tidyr::pivot_wider(names_from = attribute, values_from = value)
```

The flattened repeat group

- Two pivots, not one — The first turns every numbered column into rows; the second turns the attribute names back into...

The empty slots, and the ones that mean something — In Python

```
print(members["age"].isna().sum(), "empty slots")

members = members[members["age"].notna() | members["sex"].notna()]
```

The empty slots, and the ones that mean something — In R

```r
members |> summarise(empty = sum(is.na(age) & is.na(sex)))
```

```r
members <- members |> filter(!is.na(age) | !is.na(sex))
```

The empty slots, and the ones that mean something — In Python

```
counted = members.groupby("household_id").size().rename("members_found")
check = households.join(counted, on="household_id")
mismatch = check[check["members_found"] != check["household_size"]]
print(len(mismatch), "households where the roster does not match household_size")
```

The empty slots, and the ones that mean something — In R

```
check <- households |>
  left_join(count(members, household_id, name = "members_found"), by = "household_id") |>
  filter(members_found != household_size)
```

Long to wide: the DHIS2 pivot — In Python

```
wide = vax.pivot_table(
    index=["facility_id", "period"],
    columns="antigen",
    values="doses_administered",
    aggfunc="sum",
).reset_index()

print(len(vax), "->", len(wide))
```

Long to wide: the DHIS2 pivot — In R

```r
wide <- vax |>
  tidyr::pivot_wider(
    id_cols = c(facility_id, period),
    names_from = antigen,
    values_from = doses_administered
  )

cat(nrow(vax), "->", nrow(wide), "\n")
```

Long to wide: the DHIS2 pivot

- pandas `pivot` raises on duplicates; `pivot_table` aggregates them — Reach for `pivot` first, precisely because you...

The fill value that invents data — In Python

```
wide = vax.pivot_table(
    index=["facility_id", "period"], columns="antigen",
    values="doses_administered", aggfunc="sum", fill_value=0,
)
```

The fill value that invents data — In R

```
wide <- vax |>
  tidyr::pivot_wider(names_from = antigen, values_from = doses_administered,
                     values_fill = 0)
```

The fill value that invents data — In Python

```
wide = wide.assign(reported=wide.notna().sum(axis=1))
```

The fill value that invents data — In R

```
wide <- wide |> mutate(reported = rowSums(!is.na(across(bcg:penta3))))
```

The wide table you cannot filter — In Python

```
low = vax[vax["doses_administered"] < 10]
```

The wide table you cannot filter — In R

```
low <- vax |> filter(doses_administered < 10)
```

Round-trip, and check it — In Python

```
back = wide.melt(id_vars=["facility_id", "period"],
                 var_name="antigen", value_name="doses_administered").dropna()

assert len(back) == len(vax)
assert back["doses_administered"].sum() == vax["doses_administered"].sum()
```

Round-trip, and check it — In R

```
back <- wide |>
  tidyr::pivot_longer(-c(facility_id, period),
                      names_to = "antigen", values_to = "doses_administered") |>
  filter(!is.na(doses_administered))

stopifnot(nrow(back) == nrow(vax),
          sum(back$doses_administered) == sum(vax$doses_administered))
```

What comes next

- You can now put a table into whatever shape the question needs.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/joining-and-reshaping/joining-04-long-and-wide