One row per what?

The grain of a table decides what every number computed from it means. Households against people, episodes against patients, and the many-to-many that turns 2,403 rows into 13,004 without anyone deciding it should.

Pierre Robentz Cassion

Lesson 3 of 8

What this lesson covers

- Say the grain out loud
- The grain decides what the number means
- Expanding households to people
- Weighting is usually better than expanding
- Aggregate to the grain before you join
- The many-to-many nobody chose
- Episodes, not people
- Write the grain into the file name
- What comes next

Say the grain out loud — In Python

```python
# grain: one household interview
wash = pd.read_csv("wash-household-survey-2024.v1.csv")

# grain: one student per school day
attendance = pd.read_csv("school-attendance-2024.v1.csv")

# grain: one facility x month x antigen
vax = pd.read_csv("vaccination-coverage-2024.v1.csv")
```

Say the grain out loud — In R

```r
# grain: one household interview
wash <- read_csv("wash-household-survey-2024.v1.csv")

# grain: one student per school day
attendance <- read_csv("school-attendance-2024.v1.csv")

# grain: one facility x month x antigen
vax <- read_csv("vaccination-coverage-2024.v1.csv")
```

The grain decides what the number means — In Python

```
sized = wash[wash["household_size"].notna() & wash["litres_per_person_day"].notna()]

by_household = (sized["litres_per_person_day"] < 15).mean()
by_person = (
    sized.loc[sized["litres_per_person_day"] < 15, "household_size"].sum()
    / sized["household_size"].sum()
)

print(f"households below 15 L/p/d: {by_household:.1%}")
print(f"people in those households: {by_person:.1%}")
```

The grain decides what the number means — In R

```r
sized <- wash |> filter(!is.na(household_size), !is.na(litres_per_person_day))

sized |>
  summarise(
    by_household = mean(litres_per_person_day < 15),
    by_person = sum(household_size[litres_per_person_day < 15]) / sum(household_size)
  )
```

The grain decides what the number means

- Report the grain in the label — `share_of_households_below_15l` and `share_of_people_below_15l` cannot be confused;...

Expanding households to people — In Python

```
people = wash.loc[wash["household_size"].notna()].copy()
people["household_size"] = people["household_size"].astype(int)
people = people.loc[people.index.repeat(people["household_size"])]
people["person_index"] = people.groupby("household_id").cumcount() + 1

print(len(wash), "households ->", len(people), "people")
```

Expanding households to people — In R

```
people <- wash |>
  filter(!is.na(household_size)) |>
  tidyr::uncount(household_size, .remove = FALSE, .id = "person_index")

cat(nrow(wash), "households ->", nrow(people), "people\n")
```

Expanding households to people

- **It is a claim, not a transformation.** Every person in a household is now assumed to have the household's water...
- **Nothing about the individuals is real.** `person_index` is a position, not a person. Do not disaggregate by it, and...
- **The forty-six households with no recorded size vanish.** State that, or the expanded population is quietly 46...

Weighting is usually better than expanding — In Python

```
sized["weight"] = sized["household_size"]

weighted_share = (
    (sized["litres_per_person_day"] < 15) * sized["weight"]
).sum() / sized["weight"].sum()
```

Weighting is usually better than expanding — In R

```
sized |>
  summarise(share = weighted.mean(litres_per_person_day < 15, w = household_size))
```

Aggregate to the grain before you join — In Python

```python
per_student = (
    attendance.assign(present=attendance["present"].map({"true": True, "false": False}))
    .groupby("student_id")
    .agg(days_marked=("present", "size"),
         days_present=("present", "sum"))
    .reset_index()
)
per_student["attendance_rate"] = per_student["days_present"] / per_student["days_marked"]

# grain: one student. Now the join is one-to-one.
summary = roster.merge(per_student, on="student_id", how="left", validate="one_to_one")
```

Aggregate to the grain before you join — In R

```r
per_student <- attendance |>
  summarise(
    days_marked  = sum(present %in% c(TRUE, FALSE)),
    days_present = sum(present %in% TRUE),
    .by = student_id
  ) |>
  mutate(attendance_rate = days_present / days_marked)

summary <- roster |>
  left_join(per_student, by = "student_id", relationship = "one-to-one")
```

The many-to-many nobody chose — In Python

```
# household file: one row per household. member file: one row per person.
# Both carry `community`. Joining on it instead of household_id:
bad = households.merge(members, on="community", how="inner")
```

The many-to-many nobody chose — In R

```r
bad <- households |> inner_join(members, by = "community")
```

Episodes, not people

- **Caseload** is a count of episodes. Two admissions of the same child are two admissions, and reporting them as one...
- **Children reached** is a count of distinct children, which is smaller.
- **Cured rate** has episodes in both numerator and denominator, and mixing an episode numerator with a child denominator...

Episodes, not people — In Python

```
print("episodes:", len(admissions))
print("children:", admissions["child_id"].nunique())
print("readmitted:", (admissions.groupby("child_id").size() > 1).sum())
```

Episodes, not people — In R

```
c(episodes = nrow(admissions),
  children = n_distinct(admissions$child_id),
  readmitted = sum(count(admissions, child_id)$n > 1))
```

Write the grain into the file name — Example

```
outputs/tables/attendance_by_student.csv
outputs/tables/attendance_by_school_month.csv
outputs/tables/coverage_by_facility_period_antigen.csv
```

What comes next

- Grain answers what a row is.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/joining-and-reshaping/joining-03-grain-and-rosters