Attaching a population denominator

Where a denominator comes from, how to join it without matching on a name, and why catchment populations do not add up to a district. Plus the coverage figure above 100% and what it is really telling you.

Pierre Robentz Cassion

Lesson 5 of 8

Joining and Reshaping Programme Data — Making the pieces line up

What this lesson covers

- The numerator is the easy half
- Where a denominator actually comes from
- Join on the code, never on the name
- The projection, and the year nobody states
- Catchment populations do not add up
- Coverage above 100%
- Rates per thousand, and the annualisation trap
- Write the denominator down
- What comes next

The numerator is the easy half

- Counting what your programme did is bookkeeping.

Where a denominator actually comes from

- **A census projection.** The national statistics office publishes a census and a growth rate; every district population...
- **An administrative population frame.** The health ministry's own denominators by facility catchment, which is what...
- **A programme target.** How many people the programme planned to reach. A legitimate denominator for "did we do what we...
- **A survey.** The most defensible and the least available, because a survey gives you a proportion with a confidence...
- Name the source in the column — `target_population_moh_2024` and `target_population_census_projection` are different...

Join on the code, never on the name — In Python

```
population = pd.read_csv("admin-population-2024.csv")   # admin2_pcode, year, pop_total,
    pop_under1

vax["admin2_pcode"] = vax["admin2_pcode"].astype("string").str.strip().str.upper()

with_denominator = vax.merge(
    population.query("year == 2024"),
    on="admin2_pcode", how="left", validate="many_to_one",
)

missing = with_denominator["pop_under1"].isna().sum()
assert missing == 0, f"{missing} rows have no population figure"
```

Join on the code, never on the name — In R

```r
population <- read_csv("admin-population-2024.csv")

with_denominator <- vax |>
  mutate(admin2_pcode = toupper(stringr::str_squish(admin2_pcode))) |>
  left_join(filter(population, year == 2024),
            by = "admin2_pcode", relationship = "many-to-one")

stopifnot(!any(is.na(with_denominator$pop_under1)))
```

The projection, and the year nobody states — In Python

```
GROWTH = 0.024   # national annual growth rate, published with the census
CENSUS_YEAR = 2015

population["pop_2024"] = population["pop_census"] * (1 + GROWTH) ** (2024 - CENSUS_YEAR)
```

The projection, and the year nobody states — In R

```
GROWTH <- 0.024
CENSUS_YEAR <- 2015

population <- population |>
  mutate(pop_2024 = pop_census * (1 + GROWTH)^(2024 - CENSUS_YEAR))
```

The projection, and the year nobody states

- Write the census year, the growth rate and its source next to the number.
- Use the **same** projection for every indicator in a report. Two indicators on two projections cannot be compared, and…
- Where displacement has happened, say that the projection does not account for it, and give the direction of the error…

Catchment populations do not add up — In Python

```
by_facility = vax.query("antigen == 'penta3' and period == '2024-01-01'")
print("sum of facility targets:", by_facility["target_population"].sum())
```

Catchment populations do not add up — In R

```
vax |>
  filter(antigen == "penta3", period == "2024-01-01") |>
  summarise(total = sum(target_population))
```

Catchment populations do not add up — In Python

```
district = (
    vax.query("antigen == 'penta3'")
    .groupby("period")
    .agg(doses=("doses_administered", "sum"),
         target=("target_population", "sum"))
)
district["coverage"] = district["doses"] / district["target"]
```

Catchment populations do not add up — In R

```
district <- vax |>
  filter(antigen == "penta3") |>
  summarise(doses = sum(doses_administered),
            target = sum(target_population), .by = period) |>
  mutate(coverage = doses / target)
```

Coverage above 100%

- **The denominator is too small.** An out-of-date projection, or a catchment coefficient that assumes fewer under-ones…
- **The numerator includes people from outside.** Boundary crossing, or a campaign that drew people in from a…
- **The grain is wrong.** Doses counted where children were meant — a child receiving three doses of a three-dose antigen…
- **A join multiplied something.** Everything in lesson 1.

Coverage above 100% — In Python

```
over = district[district["coverage"] > 1]
print(over)
```

Coverage above 100% — In R

```
district |> filter(coverage > 1)
```

Coverage above 100%

- Report it rather than capping it — A coverage figure clipped at 100% has thrown away the evidence that the denominator...

Rates per thousand, and the annualisation trap — In Python

```
district["per_1000"] = 1000 * district["doses"] / district["target"]
```

Rates per thousand, and the annualisation trap — In R

```
district <- district |> mutate(per_1000 = 1000 * doses / target)
```

Write the denominator down

Field	Value
Indicator	Penta3 coverage, district
Numerator	Penta3 doses administered, facilities reporting
Denominator	Surviving infants, MoH catchment figures summed to district
Source	MoH population estimates 2024, projected from 2015 census at 2.4%
Disaggregation	Month, facility type
Caveat	Reporting rate 29% in August; coverage computed on reporting facilities only

What comes next

- The denominator answers "out of how many people".

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/joining-and-reshaping/joining-05-population-denominators