When the register and the report disagree

Recompute the indicator from the line-level register, put it beside what was reported upward, and decompose the difference by reporting unit instead of averaging it away.

Pierre Robentz Cassion

Lesson 7 of 8

Joining and Reshaping Programme Data — The table you analyse

What this lesson covers

- Two numbers for the same thing
- Recompute from the register
- The coding decision that moves a denominator
- Put the two sources in one table
- Decompose, do not average
- Why two sources disagree
- Report the gap; do not reconcile it away
- What comes next

Two numbers for the same thing

- Somewhere there is a line-level register — daily attendance marks, a screening book, a patient register.

Recompute from the register — In Python

```
attendance["marked"] = attendance["present"].isin(["true", "false"])
attendance["is_present"] = attendance["present"] == "true"

from_register = (
    attendance.merge(roster[["student_id", "school_id"]], on="student_id",
                     how="left", validate="many_to_one")
    .groupby("school_id")
    .agg(marks=("marked", "sum"), present=("is_present", "sum"))
)
from_register["rate"] = from_register["present"] / from_register["marks"]
```

Recompute from the register — In R

```r
from_register <- attendance |>
  left_join(select(roster, student_id, school_id), by = "student_id",
            relationship = "many-to-one") |>
  summarise(
    marks   = sum(present %in% c("true", "false")),
    present = sum(present == "true", na.rm = TRUE),
    .by = school_id
  ) |>
  mutate(rate = present / marks)
```

The coding decision that moves a denominator — In Python

```
mapping = {"true": True, "false": False, "Y": True, "N": False}
attendance["is_present_mapped"] = attendance["present"].map(mapping)

both_ways = (
    attendance.merge(roster[["student_id", "school_id"]], on="student_id")
    .groupby("school_id")
    .agg(
        marks_strict=("is_present", lambda s: s.notna().sum()),
        rate_strict=("is_present", "mean"),
        marks_mapped=("is_present_mapped", lambda s: s.notna().sum()),
        rate_mapped=("is_present_mapped", "mean"),
    )
)
print(both_ways.loc["SCH09"])
```

The coding decision that moves a denominator — In R

```r
both_ways <- attendance |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  mutate(mapped = dplyr::recode(present, "Y" = "true", "N" = "false")) |>
  summarise(
    marks_strict = sum(present %in% c("true", "false")),
    rate_strict  = mean(present[present %in% c("true", "false")] == "true"),
    marks_mapped = sum(mapped %in% c("true", "false")),
    rate_mapped  = mean(mapped[mapped %in% c("true", "false")] == "true"),
    .by = school_id
  )
```

The coding decision that moves a denominator

SCH09	Strict	Y/N mapped
Marks in denominator	2,015	2,865
Attendance rate	86.05%	85.72%

Put the two sources in one table — In Python

```
comparison = (
    from_register.rename(columns={"present": "register_present"})
    .join(reported.rename(columns={"present": "reported_present"}), how="outer")
)
comparison["difference"] = comparison["register_present"] - comparison["reported_present"]
comparison["verification_factor"] = (
    comparison["register_present"] / comparison["reported_present"]
)
print(comparison.sort_values("difference").head(10))
```

Put the two sources in one table — In R

```
comparison <- from_register |>
  full_join(reported, by = c("school_id", "month"),
            suffix = c("_register", "_reported")) |>
  mutate(
    difference = present_register - present_reported,
    verification_factor = present_register / present_reported
  ) |>
  arrange(difference)
```

Decompose, do not average — In Python

```
print(comparison["verification_factor"].describe())
print(comparison[comparison["verification_factor"].sub(1).abs() > 0.05])
```

Decompose, do not average — In R

```
comparison |>
  summarise(across(verification_factor, list(min = min, median = median, max = max)))

comparison |> filter(abs(verification_factor - 1) > 0.05)
```

Decompose, do not average

- Report the distribution and the outliers, never the mean alone — A tolerance band — anything outside 0.95 to 1.05 gets...

Why two sources disagree

- **Different definitions.** The register counts marks; the report counts enrolled children. The most common cause by a…
- **Different periods.** The report covers a calendar month, the register was extracted on the 28th. Check the boundaries…
- **Different coverage.** The report includes a school the register extract excluded, or vice versa. The anti-join from…
- **Transcription.** Somebody added the column up by hand. This is the cause everyone assumes and the least frequent of…

Report the gap; do not reconcile it away — In Python

```
summary = pd.DataFrame({
    "source": ["Line-level register", "Reported monthly returns"],
    "attendance days recorded": [69101, 68240],
    "attendance rate": ["88.5%", "89.1%"],
})
```

Report the gap; do not reconcile it away — In R

```r
summary <- tibble::tibble(
  source = c("Line-level register", "Reported monthly returns"),
  days   = c(69101, 68240),
  rate   = c("88.5%", "89.1%")
)
```

Report the gap; do not reconcile it away

A gap you report is a finding. A gap you close quietly is a figure nobody can reproduce, including you, in six months.

What comes next

- Every piece is now in place: joins you can prove, a stated grain, the right shape, a denominator, a complete calendar and a reconciled source.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/joining-and-reshaping/joining-07-register-vs-aggregate