Twelve hundred students, twenty-four decisions

The feeding programme was assigned by school, not by child. Test it on 1,200 students and t is 5.41; test it on the 24 schools that were actually assigned and t is 3.11. The effect is the same size. Only one of the two standard errors is honest.

Pierre Robentz Cassion

Lesson 6 of 8

Applied Statistics for Programmes — Where comparisons break

What this lesson covers

- Check who was assigned before you check who was measured
- The two analyses
- Why the ratio is 1.8
- Three ways to get it right
- The same trap in three other courses
- Read the ratio as a claim about generalisation
- Report it whole
- What comes next

Check who was assigned before you check who was measured — In Python

```
import pandas as pd
import numpy as np
from scipy import stats

roster = pd.read_csv("school-roster-2024.v1.csv")
print(f"roster rows {len(roster)}, students {roster['student_id'].nunique()}")

# Two students were transferred and never de-registered, so they appear twice
# with different schools. Keep the later row: that is the school they moved to.
roster = roster.drop_duplicates("student_id", keep="last")

mixed = roster.groupby("school_id")["feeding_programme"].nunique()
print(f"schools with both fed and unfed students: {(mixed > 1).sum()}")
print(roster.groupby("school_id")["feeding_programme"].first().value_counts())
```

Check who was assigned before you check who was measured — In R

```
library(dplyr)

roster |> summarise(variants = n_distinct(feeding_programme), .by = school_id) |>
  count(variants)
```

Check who was assigned before you check who was measured

- Not one school of the twenty-four has both fed and unfed students — Fifteen schools run the programme and nine do not,...
- The roster has 1,202 rows for 1,200 students — and that has to be settled first: two children were transferred without...
- Run this check before every comparison — It takes one line, and it is the difference between a t of 5.41 and a t of 3.11

The two analyses — In Python

```
attendance = pd.read_csv("school-attendance-2024.v1.csv")
MARKS = {"true": True, "Y": True, "false": False, "N": False}
marked = attendance[attendance["present"].isin(MARKS)].copy()
marked["attended"] = marked["present"].map(MARKS)

per_student = (marked.groupby("student_id")["attended"].mean()
               .rename("rate").reset_index()
               .merge(roster, on="student_id"))

fed = per_student[per_student["feeding_programme"]]["rate"]
unfed = per_student[~per_student["feeding_programme"]]["rate"]
print(stats.ttest_ind(fed, unfed, equal_var=False))
```

The two analyses — In R

```
per_student |>
  t.test(rate ~ feeding_programme, data = _)
```

The two analyses

- Student level: 90.14% against 85.21%, difference 4.93 points, t = 5.41 on 1,200 students

The two analyses — In Python

```
per_school = (per_student.groupby(["school_id", "feeding_programme"])["rate"]
              .mean().reset_index())

fed_s = per_school[per_school["feeding_programme"]]["rate"]
unfed_s = per_school[~per_school["feeding_programme"]]["rate"]
result = stats.ttest_ind(fed_s, unfed_s, equal_var=False)
print(f"n = {len(fed_s)} fed, {len(unfed_s)} unfed")
print(f"difference {fed_s.mean() - unfed_s.mean():+.2%}, t = {result.statistic:.2f}")
```

The two analyses — In R

```
per_school |> t.test(rate ~ feeding_programme, data = _)
```

The two analyses

- School level: 90.25% against 85.04%, difference 5.21 points, t = 3.11 on 24 schools

The two analyses

Analysis	n	Difference	Standard error	t
Student level	1,200	+4.93 pts	0.91 pts	**5.41**
School level	**24**	**+5.21 pts**	**1.68 pts**	**3.11**

The two analyses

- The effect barely moved. The standard error nearly doubled — That is the whole phenomenon: treating clustered...

Why the ratio is 1.8 — In Python

```
groups = [g["rate"].values for _, g in per_student.groupby("school_id")]
k, N = len(groups), len(per_student)
grand = per_student["rate"].mean()

msb = sum(len(g) * (g.mean() - grand) ** 2 for g in groups) / (k - 1)
msw = sum(((g - g.mean()) ** 2).sum() for g in groups) / (N - k)
n0 = (N - sum(len(g) ** 2 for g in groups) / N) / (k - 1)

icc = (msb - msw) / (msb + (n0 - 1) * msw)
deff = 1 + (n0 - 1) * icc
print(f"ICC {icc:.3f}, average cluster {n0:.0f}, design effect {deff:.2f}")
print(f"effective sample size {N / deff:.0f} of {N}")
```

Why the ratio is 1.8 — In R

```
# lme4::lmer(rate ~ 1 + (1 | school_id)) gives the same variance components.
```

Why the ratio is 1.8

- ICC 0.060, average cluster 50, design effect 3.94 — Six per cent of the variation in attendance is between schools,...
- Effective sample size 304, not 1,200 — The standard error grows by the square root of the design effect, and the square...

Three ways to get it right

- Aggregate to the unit of assignment — Compute one number per school, test the
- Simple, transparent, and it is what the table above does. The cost is that a

Three ways to get it right — In Python

```
sizes = per_student.groupby("school_id").size().rename("students")
sized = per_school.merge(sizes, on="school_id")
fed_rows = sized[sized["feeding_programme"]]
print(f"unweighted {fed_rows['rate'].mean():.2%}, "
      f"weighted by roster {np.average(fed_rows['rate'], weights=fed_rows['students']):.2%}")
```

Three ways to get it right — In R

```
# Weighting is a judgement about what the average school means, not a fix.
```

Three ways to get it right

- Use a mixed model with a random intercept for school — Keeps every student, estimates the between-school variance. . .
- Use cluster-robust standard errors — Keeps the student-level model and corrects the standard error for clustering,. . .
- With 24 clusters, aggregate — The mixed model buys precision when there are many clusters of unequal size; here it. . .

The same trap in three other courses

- Correlation — The point-biserial correlation between feeding and attendance is 0.161 at student level and 0.588 at...

The same trap in three other courses — In Python

```
print(f"student level r = {np.corrcoef(per_student['feeding_programme'], per_student['rate'])
    [0,1]:.3f}")
print(f"school level  r = {np.corrcoef(per_school['feeding_programme'], per_school['rate'])
    [0,1]:.3f}")
```

The same trap in three other courses — In R

```
cor(as.numeric(per_student$feeding_programme), per_student$rate)
cor(as.numeric(per_school$feeding_programme), per_school$rate)
```

The same trap in three other courses

- Water points — Functionality is assigned by point and measured by visit
- Referral cases — A case worker with forty cases is one worker, and comparing outcomes across case workers on 1,600...
- Repeated measures on the same household — Three survey rounds on one household are three rows and one household, and...

Read the ratio as a claim about generalisation

- With 24 schools, "does school feeding raise attendance" is being answered by fifteen schools that have it against nine. . .
- The student-level t of 5.41 is a claim about a study that was never run — It is the answer you would get if 1,200. . .

Report it whole — Example

```
Attendance and school feeding, February to April 2024

  Schools with feeding      90.25%   15 schools
  Schools without           85.04%    9 schools
  Difference +5.21 points, 95% CI 1.6 to 8.8, t = 3.11, Welch df 12.7

  The programme is assigned by school: no school has both fed and unfed
  students, so the analysis has 24 independent units and not 1,200. The
  student-level comparison gives the same effect with t = 5.41; that
  statistic is not reported because it treats 50 children in one school as
  50 independent observations (ICC 0.060, design effect 3.94, effective
  sample size 304).

  The comparison is observational. Schools were not randomised into the
  programme and the difference should not be read as the effect of feeding
  alone.
```

Report it whole

- The last paragraph costs two lines and is the one a reviewer will look for — Getting the unit of analysis right makes...

What comes next

- The next lesson is about correlation — where the same clustering problem changes an r from 0.16 to 0.59, and where the sentence written under the number does most of the damage.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/applied-statistics-for-programmes/stat-06-the-unit-of-analysis