Twenty-four tests, two findings, no effect

The enrolment register draws sex independently of everything else, so the over-age gap by sex is exactly zero by construction. Test it school by school and two of the twenty-four come back significant, which is roughly what chance owed you.

Pierre Robentz Cassion

Lesson 5 of 8

What this lesson covers

- A comparison with a known answer
- Two is what chance owed you
- Bonferroni: divide the threshold by the number of tests
- Benjamini–Hochberg, when twenty-four becomes two hundred
- Correction does not destroy real findings
- Count the tests, including the ones you did not report
- Report it whole
- What comes next

A comparison with a known answer — In Python (cont.)

```
import pandas as pd
import numpy as np
from statsmodels.stats.proportion import proportions_ztest

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
clean = enrolment[(enrolment["school_year"] == 2024)
                  & (enrolment["age_years"] <= 20)
                  & enrolment["grade"].between(1, 6)]
clean = clean.assign(over_age=clean["age_years"] > clean["grade"] + 5)

results = []
for school, group in clean.groupby("school_id"):
    counts = group.groupby("sex")["over_age"].agg(["sum", "size"])
    if counts["size"].min() < 5:
        continue
    stat, p = proportions_ztest(counts["sum"], counts["size"])
```

A comparison with a known answer — In Python (cont.)

```
    results.append({"school": school, "p": p,
                    "boys": counts.loc["m", "sum"] / counts.loc["m", "size"],
                    "girls": counts.loc["f", "sum"] / counts.loc["f", "size"]})

tests = pd.DataFrame(results).sort_values("p")
print(f"{len(tests)} schools tested, {(tests['p'] < 0.05).sum()} significant")
```

A comparison with a known answer — In R

```r
library(dplyr)

clean |>
  summarise(k = sum(over_age), n = n(), .by = c(school_id, sex)) |>
  tidyr::pivot_wider(names_from = sex, values_from = c(k, n)) |>
  filter(n_m >= 5, n_f >= 5) |>
  rowwise() |>
  mutate(p = prop.test(c(k_m, k_f), c(n_m, n_f))$p.value)
```

A comparison with a known answer

School	Boys	Girls	p
SCH23	62.5% (10/16)	25.9% (7/27)	**0.018**
SCH06	39.1% (9/23)	12.5% (3/24)	**0.036**
SCH21	43.5% (10/23)	25.8% (8/31)	0.173
SCH03	50.0% (11/22)	30.8% (8/26)	0.175
SCH04	27.3% (3/11)	55.6% (5/9)	0.199

A comparison with a known answer

- Twenty-four schools tested, two significant at $p < 0.05$ — Both would be written up
- Neither is real — The generator never let sex touch anything

Two is what chance owed you — In Python

```
n_tests = len(tests)
print(f"tests run: {n_tests}")
print(f"expected significant if nothing is real: {0.05 * n_tests:.1f}")
print(f"chance of at least one: {1 - 0.95 ** n_tests:.1%}")
```

Two is what chance owed you — In R

```
# 1 - 0.95^24. The arithmetic is one line and it is the whole lesson.
```

Two is what chance owed you

- Expected false positives: 1.2. Observed: 2 — The probability of getting *at least one* significant result from...
- A p-value threshold of 0.05 is a promise about one test — It says that if nothing is going on, you will be misled one...
- This is why "we looked at it by school and two schools stood out" is not a finding — It is a description of what...

Bonferroni: divide the threshold by the number of tests — In Python

```
BONFERRONI = 0.05 / n_tests
print(f"corrected threshold: {BONFERRONI:.5f}")
print(f"surviving: {(tests['p'] < BONFERRONI).sum()}")
```

Bonferroni: divide the threshold by the number of tests — In R

```
p.adjust(tests$p, method = "bonferroni") < 0.05
```

Bonferroni: divide the threshold by the number of tests

- Threshold $0.05/24 = 0.00208$. Surviving: none — The smallest p among the twenty-four is 0.018, an order of magnitude away

Benjamini–Hochberg, when twenty-four becomes two hundred — In Python

```
from statsmodels.stats.multitest import multipletests

reject, adjusted, _, _ = multipletests(tests["p"], alpha=0.05, method="fdr_bh")
print(f"BH survivors: {reject.sum()}")
```

Benjamini–Hochberg, when twenty-four becomes two hundred — In R

```r
p.adjust(tests$p, method = "BH") < 0.05
```

Benjamini–Hochberg, when twenty-four becomes two hundred

- Benjamini–Hochberg also rejects none of the twenty-four — which is the correct answer here — there is nothing to find
- Use Bonferroni for a small confirmatory family; use BH for a large exploratory screen — Both are one line, and the...

Correction does not destroy real findings — In Python

```
previous = enrolment[enrolment["school_year"] == 2023].set_index("student_id")
clean = clean.assign(last_year=clean["student_id"].map(previous["end_of_year_status"]))
returning = clean[clean["last_year"].isin(["repeated", "promoted"])]

survivors = []
for school, group in returning.groupby("school_id"):
    counts = group.groupby("last_year")["over_age"].agg(["sum", "size"])
    if len(counts) < 2 or counts["size"].min() < 5:
        continue
    stat, p = proportions_ztest(counts["sum"], counts["size"])
    survivors.append(p)

survivors = np.array(survivors)
print(f"{len(survivors)} schools, {(survivors < 0.05).sum()} significant, "
      f"{(survivors < 0.05 / len(survivors)).sum()} survive Bonferroni")
```

Correction does not destroy real findings — In R

```
# Same loop, different comparison. The generator made this one real.
```

Correction does not destroy real findings

Comparison	Tests	Significant at 0.05	Survive Bonferroni
Over-age by **sex** (no true effect)	24	2	**0**
Over-age by **2023 repetition** (real)	14	14	**9**

Correction does not destroy real findings

- The 94.5% versus 30.6% gap from the education course survives correction in nine of the fourteen schools where it can...

Count the tests, including the ones you did not report

- Subgroups you looked at and dropped — Splitting by district, seeing nothing, and splitting by grade instead is two...
- Outcomes you tried in turn — Testing over-age, then attendance, then completion against the same grouping is three tests
- Cut points you moved — "Over-age" at grade + 2 rather than grade + 1 is a second test of the same idea
- The analysis someone else ran on the same data — Rare to know, and worth asking about when a reviewer arrives with a...

Count the tests, including the ones you did not report — In Python

```
plan = {
    "outcomes": ["over_age"],
    "groupings": ["sex"],
    "subgroups": ["school_id"],
}
n_planned = 1 * 1 * 24
print(f"tests declared in the analysis plan: {n_planned}")
```

Count the tests, including the ones you did not report — In R

```
# Write the number down before running anything. It cannot be recovered after.
```

Count the tests, including the ones you did not report

- Declare the family before you run it — After the fact, the number of tests is a matter of memory and self-interest, and...

Report it whole — Example

```
Over-age enrolment by sex, school-level analysis

  24 schools tested (both sexes with at least 5 students in grades 1-6).
  2 schools significant at p < 0.05: SCH23 (p = 0.018), SCH06 (p = 0.036).

  With 24 tests, 1.2 significant results are expected by chance alone and
  the probability of at least one is 71%. Neither school survives the
  Bonferroni threshold of 0.0021, and neither is reported as a finding.

  For comparison, over-age by 2023 repetition is significant in all 14
  schools testable and survives Bonferroni in 9 of them.
```

Report it whole

- The comparison row is what makes the block persuasive rather than defensive — It shows the same procedure finding...

What comes next

- Every test so far has assumed one row is one independent observation.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/

applied-statistics-for-programmes/stat-05-twenty-four-tests