CASSION · COURSE

Applied Statistics for Programmes

Uncertainty, comparison and significance, taught around the claims a programme report makes — not around a textbook sequence of tests.

Instructor	Pierre Robentz Cassion
Level	Intermediate
Learner hours	24 h
Taught in	Python · R
Updated	July 28, 2026
Sectors	Public Health · Nutrition · WASH · Education · Protection
Standards and methodologies	SMART survey · Demographic and Health Survey (DHS) · UNICEF indicator definitions · OECD DAC evaluation criteria

Read and run the course online at data-analysis.cassion.dev/courses/applied-statistics-for-programmes

What you will be able to do

- Read the distribution of an indicator before computing anything on it, and say what its mean is hiding
- Put a confidence interval on a proportion and write the result as an interval in a report sentence
- Compare two groups with the right test, state its assumptions, and check them against the data
- Separate statistical significance from operational relevance, and report an effect size beside every p-value
- Recognise when the unit of analysis is not the row, and correct the sample size accordingly
- Run a family of comparisons without manufacturing the false positives it invites

Prerequisites

None. This course assumes no prior programming experience.

Contents

About this course

Module 4 spent six courses deferring to this one. Every time a lesson said "put an interval on that before writing the sentence", it was pointing here — and the gaps it deferred are the worked examples, already computed on files you have already cleaned.

This is not a statistics textbook with programme data as decoration. It is organised around the four claims a programme report actually makes: *this is the level*, *this group differs from that one*, *this changed*, and *this caused that*. Each unit takes one and works out what it costs to say it honestly.

Three results shape the course, all computed from committed files.

The mean E. coli count in tested households is 29.6 CFU/100 mL and the median is 0. More than half of tested households have no detectable contamination and the mean is describing a tail that reaches 608. A report quoting the mean has described no household in the sample.

Two gaps from module 4, one real and one not. Boys are over-age at 39.2% against girls at 33.8% — a 5.5-point gap whose interval runs from −0.4 to +11.3, so the honest answer is that this survey cannot tell. Cases reporting a disability complete a referral at 26.7% against 46.2% — a 19.4-point gap with an interval from −26.1 to −12.8, which no reasonable reading makes zero.

1,200 students are 24 schools. Not one of the twenty-four schools has both fed and unfed students, so a comparison treating children as independent observations has an effective sample size of 24 and a standard error 1.8 times too small.

Both Python and R throughout. You should have done modules 3 and 4 — this course assumes you can defend a denominator, and it spends its time on what to do once you have one.

Part I

Look at it first

CHAPTER 1

A mean of 29.6 and a median of zero

Lesson 1 of 8 · 180 min

The mean E. coli count in tested households is 29.6 CFU/100 mL. The median is 0. More than half the sample has no detectable contamination, and the mean is describing a tail that reaches 608.

1.1 Compute the summary last

```python
import pandas as pd

wash = pd.read_csv("wash-household-survey-2024.v1.csv")
ecoli = wash["ecoli_cfu_100ml"].dropna()

print(ecoli.describe().round(1))
```

```r
library(dplyr)

wash |> filter(!is.na(ecoli_cfu_100ml)) |>
  summarise(n = n(), mean = mean(ecoli_cfu_100ml),
            median = median(ecoli_cfu_100ml), sd = sd(ecoli_cfu_100ml))
```

	Value
n	802
Mean	**29.6**
Median	**0.0**
Standard deviation	85.6
Maximum	608

The mean is 29.6 and the median is 0. Those two numbers describe the same 802 households, and only one of them describes any of them.

More than half the tested households have no detectable E. coli at all. The mean is what you get by averaging a large group of zeros with a small group of very large numbers, and it lands in a region where almost nobody sits.

1.2 Look at the shape

```python
bins = [-1, 0, 10, 100, 1000]
labels = ["0 (none detected)", "1-10", "11-100", ">100"]
print(pd.cut(ecoli, bins, labels=labels).value_counts().reindex(labels))
print(f"\nskew: {ecoli.skew():.2f}")
```

```r
wash |> filter(!is.na(ecoli_cfu_100ml)) |>
  count(band = cut(ecoli_cfu_100ml, c(-1, 0, 10, 100, Inf)))
```

Band	Households	Share
0 — none detected	430	**53.6%**
1–10	173	21.6%
11–100	139	17.3%
>100	60	7.5%

Skew +4.04. A symmetric distribution has a skew near zero; anything past about +1 means the mean and the median are answering different questions.

This is why the WASH course reported risk classes rather than a mean. The classes are not a presentational choice — they are the only summary that survives a distribution shaped like this one.

1.3 Three shapes and what each demands

Run the same three numbers over four indicators from four courses and the pattern is immediate.

```
import numpy as np

def profile(series, name):
    s = series.dropna()
    return {
        "indicator": name, "n": len(s),
        "mean": round(s.mean(), 1), "median": round(s.median(), 1),
        "skew": round(s.skew(), 2),
    }

points = pd.read_csv("water-point-monitoring-2024.v1.csv")
protection = pd.read_csv("protection-referrals-2024.v1.csv")

print(pd.DataFrame([
    profile(wash["ecoli_cfu_100ml"], "E. coli, CFU/100mL"),
    profile(points["days_since_breakdown"], "days a water point is down"),
    profile(protection["days_to_first_service"], "days to first service"),
    profile(wash["litres_per_person_day"], "litres per person per day"),
]))
```

```
# One function, four indicators, four different answers about which summary
# to report.
```

Indicator	n	Mean	Median	Skew
E. coli, CFU/100 mL	802	29.6	**0.0**	+4.04
Days a water point is down	709	125.7	56.0	+1.64
Days to first service	728	10.1	9.0	+0.52
Litres per person per day	2,403	24.2	23.7	**+8.54**

Days to first service is nearly symmetric — mean 10.1, median 9.0, skew +0.52. A mean is a fair summary and a standard deviation means something.

Days a water point is down is heavily right-skewed — a mean of 125.7 against a median of 56, because a handful of abandoned points have been broken for over 600 days. Report the median and the quartiles.

Litres per person per day looks almost symmetric and has a skew of +8.54. That combination is the interesting one.

1.4 When the skew is the data error

```python
litres = wash["litres_per_person_day"].dropna()
print(f"median {litres.median():.1f}, p90 {litres.quantile(0.90):.1f}, "
      f"max {litres.max():.1f}")
print(f"above 80: {(litres > 80).sum()} households")
```

```r
wash |> summarise(median = median(litres_per_person_day),
                  p90 = quantile(litres_per_person_day, 0.9),
                  max = max(litres_per_person_day))
```

Median 23.7, ninetieth percentile 33.7, maximum 277.6. **The skew statistic is not describing the population; it is detecting eleven bad rows.**

Those eleven are the households whose *total* consumption was entered in a per-person column — the defect the WASH course taught. Here it arrives from the other direction: **a skew far out of line with the interquartile range is a data quality signal before it is a distributional finding.**

```python
clean = litres[litres <= 80]
print(f"after removing 11 rows: skew {clean.skew():.2f}, "
      f"mean {clean.mean():.1f}, median {clean.median():.1f}")
```

```r
# Recompute after the correction and see whether the shape was real.
```

Eleven rows out of 2,403 and the skew goes from +8.54 to +0.06. Mean 23.5, median 23.6 — a distribution that was symmetric all along, wearing a shape that belonged entirely to a unit error.

Compute the skew, then decide whether it is a finding or a bug. Both are common and they are told apart by looking at the extreme values, not by looking at the statistic.

1.5 What to report for each shape

Shape	Report	Do not report
Roughly symmetric	Mean and standard deviation	—
Right-skewed	Median and interquartile range	A mean without the median beside it
Zero-inflated	**The share at zero**, then the distribution of the rest	A mean at all
Bounded proportion	The proportion and its interval	A standard deviation

E. coli is zero-inflated, which is a distinct case from merely skewed: 53.6% of the sample sits at exactly one value, and no continuous summary describes that. The two-part report — *how many are at zero, and among the rest, how bad* — is the only honest one.

```
E. coli at point of collection, 802 households tested

  No detectable E. coli        53.6%   430 households
  Among the 372 with any detected:
    median                     13 CFU/100 mL
    interquartile range        5 to 49
    maximum                    608
```

```
Mean over all 802 households is 29.6 CFU/100 mL. It is not reported as a
summary because 53.6% of the sample sits at zero and the mean falls in a
range occupied by almost no household.
```

1.6 The histogram you should always draw

```python
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 2, figsize=(9, 3))
axes[0].hist(ecoli, bins=40)
axes[0].set_title("E. coli, raw")
axes[1].hist(np.log1p(ecoli), bins=40)
axes[1].set_title("log(1 + E. coli)")
plt.tight_layout()
```

```r
hist(wash$ecoli_cfu_100ml, breaks = 40)
hist(log1p(wash$ecoli_cfu_100ml), breaks = 40)
```

Draw it before you summarise it, every time, and do not put it in the report. The histogram is an instrument for you, not a finding for the reader — its job is to tell you which summary is honest, and once it has done that the summary goes in and the histogram stays out.

Two minutes of looking would have prevented every error in this lesson, and that is the entire argument for the habit.

1.7 What comes next

You now know which single number describes an indicator. The next lesson is about how much that number could be wrong by, and the notation that carries it into a sentence a reader can act on.

CHAPTER 2

A proportion of zero with an interval up to 24%

Lesson 2 of 8 · 180 min

Twelve children, none of them over-age. The textbook interval says 0% to 0%. The right one says 0% to 24.3%, and the difference between those two answers is whether you would act on the cell.

2.1 Why a point estimate is not a result

Every proportion in module 4 was computed from a sample — of households, of cases, of students. Another sample from the same population would have produced a different number, and a confidence interval is how much different.

```python
import pandas as pd
import numpy as np

wash = pd.read_csv("wash-household-survey-2024.v1.csv")
open_defecation = wash["sanitation_facility"].eq("open-defecation")
k, n = open_defecation.sum(), len(wash)
print(f"{k}/{n} = {k / n:.1%}")
```

```r
library(dplyr)
wash |> summarise(k = sum(sanitation_facility == "open-defecation"), n = n())
```

319 of 2,403 — 13.3%. The interval on that is narrow, because the sample is large. On smaller cells it is not, and the whole point of computing it is that you cannot tell which case you are in by looking at the percentage.

2.2 Wilson, not Wald

The formula most people learn is the Wald interval: `p ± 1.96 × sqrt(p(1-p)/n)`. It is simple, it is what a textbook shows first, and it fails exactly where you need it.

```python
def wald(k, n, z=1.96):
    p = k / n
    se = np.sqrt(p * (1 - p) / n)
    return p - z * se, p + z * se

def wilson(k, n, z=1.96):
    p = k / n
    denom = 1 + z**2 / n
    centre = (p + z**2 / (2 * n)) / denom
    half = z * np.sqrt(p * (1 - p) / n + z**2 / (4 * n**2)) / denom
    return centre - half, centre + half

for k, n in [(0, 12), (1, 20), (3, 11)]:
    print(f"{k}/{n} = {k/n:.1%}")
    print(f"   Wald   [{wald(k, n)[0]:.1%}, {wald(k, n)[1]:.1%}]")
    print(f"   Wilson [{wilson(k, n)[0]:.1%}, {wilson(k, n)[1]:.1%}]")
```

```r
# R has this built in and it is the Wilson interval by default.
prop.test(0, 12)$conf.int
binom.test(1, 20)$conf.int
```

Cell	Wald	Wilson
0 of 12	**[0.0%, 0.0%]**	[0.0%, 24.3%]
1 of 20	**[−4.6%, 14.6%]**	[0.9%, 23.6%]
3 of 11	[1.0%, 53.6%]	[9.7%, 56.6%]

Wald says a cell with no events has zero uncertainty, and it produces negative probabilities. Both are absurd and both appear in real reports, because the formula is the one people remember.

Use Wilson, or `prop.test` in R and `statsmodels.stats.proportion.proportion_confint(method="wilson"` in Python. It costs nothing and it does not break at the boundary.

2.3 Read the width, not just the bounds

```python
cases = [
    ("Open defecation", 319, 2403),
    ("Cholera case fatality", 38, 974),
    ("Cholera CFR, Nord district", 24, 381),
    ("Referral completion, disability reported", 54, 202),
    ("Over-age, grade 6", 67, 132),
]
for name, k, n in cases:
    lo, hi = wilson(k, n)
    print(f"{name:42} {k/n:6.1%}  [{lo:.1%}, {hi:.1%}]  width {hi-lo:.1%}")
```

```r
# Same five, and the width is the column to read.
```

Estimate	Value	95% interval	Width
Open defecation	13.3%	12.0–14.7%	**2.7 pts**
Cholera case fatality	3.9%	2.9–5.3%	2.5 pts
Cholera CFR, Nord	6.3%	4.3–9.2%	4.9 pts
Referral completion, disability	26.7%	21.1–33.2%	12.1 pts
Over-age, grade 6	**50.8%**	**42.3–59.1%**	**16.8 pts**

Grade 6 over-age is 50.8% and could be anywhere from 42% to 59%. The education course reported that number to one decimal place. The decimal is not wrong, it is just spurious — the second digit of 50.8 carries no information at n=132.

Two things drive the width and only one is under your control. Sample size, which the design decided; and how close the proportion is to 50%, which the world decided. A proportion near 50% has the widest interval it can have, and 6.3% in Nord is tighter than 50.8% in grade 6 despite a larger n.

2.4 Writing it into a sentence

This is the part the spine of this course is about, and it is a writing problem as much as a statistical one.

Not this: "50.8% of grade 6 students are over-age."

Nor this: "50.8% (95% CI 42.3–59.1) of grade 6 students are over-age." Correct, and a reader skips the parenthesis.

This: "Between four and six students in ten in grade 6 are over-age (50.8%, 95% CI 42.3–59.1, n=132)."

Lead with the interval in words, then give the numbers. The words are what a non-analyst reads and they carry the uncertainty; the numbers are what an analyst checks.

```python
def sentence(label, k, n):
    lo, hi = wilson(k, n)
    return (f"{label}: {k/n:.1%} (95% CI {lo:.1%} to {hi:.1%}, n={n})")

print(sentence("Grade 6 over-age", 67, 132))
```

```python
# Generate the string in code so the number and its interval cannot drift apart.
```

2.5 When the interval changes the decision

The test of whether an interval matters is whether any value inside it would lead somewhere different.

```python
lo, hi = wilson(38, 974)
threshold = 0.01                        # Sphere: cholera CFR below 1%
print(f"CFR {38/974:.1%}, interval [{lo:.1%}, {hi:.1%}]")
print(f"entire interval above the {threshold:.0%} threshold: {lo > threshold}")
```

```r
prop.test(38, 974)$conf.int
```

Cholera case fatality is 3.9% with an interval of 2.9% to 5.3%, and the Sphere threshold is 1%. Every value in the interval is above the threshold, so the conclusion — this response is failing the standard — does not depend on where in the interval the truth sits.

That is when you can act on a point estimate: when the whole interval says the same thing. Where it straddles the threshold, the honest report says the data cannot settle it, and the next lesson is a gap where exactly that happens.

2.6 Three intervals this course will not compute this way

A proportion from a clustered sample. The survey course established that a cluster design widens the interval by the design effect, and the formula above assumes simple random sampling. Applying it to cluster data understates the width, sometimes by a factor of two.

A median or a skewed mean. The intervals here are for proportions. A median needs a bootstrap or a rank-based method, and a mean on the E. coli distribution from lesson 1 needs neither — it needs a different summary.

A count with no denominator. "142 cases were reported" has no interval, because it is not an estimate of anything. It is a count of what was recorded, and the protection course spent a lesson on why.

2.7 Report it as a range

```
Over-age enrolment, grade 6

  50.8%   95% CI 42.3 to 59.1, n = 132

  Between four and six students in ten. The interval is wide because grade 6
  holds 132 students; a difference of five points against another grade would
  not be distinguishable at this sample size.
```

The last sentence is the useful one. It tells the reader in advance which comparisons this number can support, which stops them making the one it cannot.

2.8 What comes next

An interval tells you how uncertain one number is. The next lesson puts two numbers side by side and asks whether the gap between them is real — on two gaps module 4 left open, which turn out to have opposite answers.

Part II

Comparing two groups

CHAPTER 3

Two gaps, two answers

Lesson 3 of 8 · 180 min

A 5.5-point gap with an interval from −0.4 to +11.3, and a 19.4-point gap with an interval from −26.1 to −12.8. Same test, same code, opposite conclusions — and module 4 left both of them open on purpose.

3.1 The two questions module 4 deferred

The education course found boys over-age more often than girls and declined to call it a finding. The protection course found a disability gap in referral completion and called it the most important result in the dataset. Both said "compute the interval first". This is that computation.

```python
import pandas as pd
import numpy as np
from statsmodels.stats.proportion import proportions_ztest, confint_proportions_2indep

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

boys = clean[clean["sex"] == "m"]["over_age"]
girls = clean[clean["sex"] == "f"]["over_age"]
print(f"boys  {boys.sum()}/{len(boys)} = {boys.mean():.1%}")
print(f"girls {girls.sum()}/{len(girls)} = {girls.mean():.1%}")
```

```r
library(dplyr)

enrolment |>
  filter(age_years <= 20, school_year == 2024, between(grade, 1, 6)) |>
  mutate(over_age = age_years > grade + 5) |>
  summarise(k = sum(over_age), n = n(), .by = sex)
```

3.2 Gap one: the one that is not there

Group	Over-age	n	95% interval
Boys	39.2%	510	35.1–43.5%
Girls	33.8%	542	29.9–37.8%

The two intervals overlap. **Overlapping intervals are a hint and not a test** — the correct thing to compute is an interval on the *difference*.

```python
count = np.array([boys.sum(), girls.sum()])
nobs = np.array([len(boys), len(girls)])
```

```
stat, pvalue = proportions_ztest(count, nobs)
low, high = confint_proportions_2indep(count[0], nobs[0], count[1], nobs[1])
print(f"difference {boys.mean() - girls.mean():+.1%}")
print(f"95% CI [{low:+.1%}, {high:+.1%}]")
print(f"z = {stat:.2f}, p = {pvalue:.3f}")
```

```
prop.test(c(200, 183), c(510, 542))
```

Difference +5.5 points, 95% CI −0.4 to +11.3, p = 0.066.

The interval includes zero. **So the honest answer is not "there is no gap" — it is "this survey cannot tell you whether there is a gap, and if there is one it is somewhere between girls being slightly worse off and boys being eleven points worse off".**

That is a genuinely useful sentence and it is not the same as a null result.

3.3 Gap two: the one that is

```
protection = pd.read_csv("protection-referrals-2024.v1.csv")
consenting = protection[protection["consent_to_refer"]]
reached = (consenting["referral_accepted"]
           & consenting["days_to_first_service"].notna())

disability = consenting["disability_reported"].isin([True, "true", "Yes"])
k = np.array([reached[disability].sum(), reached[~disability].sum()])
n = np.array([disability.sum(), (~disability).sum()])

stat, pvalue = proportions_ztest(k, n)
low, high = confint_proportions_2indep(k[0], n[0], k[1], n[1])
print(f"{k[0]}/{n[0]} = {k[0]/n[0]:.1%}   {k[1]}/{n[1]} = {k[1]/n[1]:.1%}")
print(f"difference {k[0]/n[0] - k[1]/n[1]:+.1%}, CI [{low:+.1%}, {high:+.1%}]")
print(f"z = {stat:.2f}, p = {pvalue:.2e}")
```

```
prop.test(c(54, 663), c(202, 1436))
```

Group	Completion	n
Disability reported	26.7%	202
Not reported	46.2%	1,436

Difference −19.4 points, 95% CI −26.1 to −12.8, p < 0.001.

Every value in that interval is a substantial gap. The smallest gap consistent with the data is 12.8 points, which is still large enough to act on. That is what makes this a finding rather than a hint.

3.4 The two results side by side

```
results = pd.DataFrame([
    {"comparison": "over-age, boys vs girls", "diff": 5.5,
     "ci": "-0.4 to +11.3", "n": "510 vs 542", "p": 0.066},
    {"comparison": "completion, disability", "diff": -19.4,
     "ci": "-26.1 to -12.8", "n": "202 vs 1436", "p": 0.0000002},
```

```
])
print(results)
```

```
# Two rows. The conclusion column is the analyst's, not the test's.
```

Notice that the significant result has the smaller sample in one arm. 202 cases produced a decisive answer and 510 students did not, because the effect size is nearly four times larger. **Sample size and effect size trade off**, and it is the combination that determines whether a comparison resolves.

3.5 Choosing the test

Three tests cover almost everything a programme report compares, and the choice is made by what the outcome is rather than by what feels sophisticated.

Comparing	Test	In this course
Two proportions	Two-sample z-test, or `prop.test`	Both gaps above
Two means	Welch's t-test	Attendance by feeding, lesson 6
A categorical against a categorical	Chi-square	Closure reason by district

```
from scipy import stats

points = pd.read_csv("water-point-monitoring-2024.v1.csv")
table = pd.crosstab(points["admin2"], points["functional_status"])
chi2, p, dof, expected = stats.chi2_contingency(table)
print(f"chi-square {chi2:.1f}, dof {dof}, p = {p:.4f}")
print(f"smallest expected cell: {expected.min():.1f}")
```

```
chisq.test(table(points$admin2, points$functional_status))
```

3.6 Check the assumptions, and say you did

Each test assumes things, and two of them are checkable in one line.

Independence. Every test above assumes each row is an independent observation. It is the assumption most often violated and the hardest to see — lesson 6 is entirely about a case where it fails.

Expected cell counts for chi-square. The test is unreliable when an expected cell falls below about five. Print `expected.min()` every time; if it is small, collapse categories or use Fisher's exact test.

Equal variances for the t-test. Do not check it — **use Welch's t-test always**, which does not assume it. `scipy.stats.ttest_ind(..., equal_var=False)` and R's `t.test` default. The version that assumes equal variances buys nothing and fails when the groups differ in spread.

```
girls_rate, boys_rate = girls.mean(), boys.mean()
print("assumptions checked:")
print(f"  independence: one row per student, no student in two rows -- verified")
print(f"  sample sizes: {len(boys)} and {len(girls)}, both well above 30")
```

```
print(f"  expected counts: smallest is {min(len(boys), len(girls)) * min(girls_rate, 1 -
    boys_rate):.0f}")


# Write the check into the script, not into your memory of having done it.
```

3.7 Report the difference, not the two numbers

```
Over-age enrolment by sex, primary, 2024

  Boys    39.2%   n = 510
  Girls   33.8%   n = 542
  Difference +5.5 points, 95% CI -0.4 to +11.3, p = 0.066

  The interval includes zero. This survey cannot establish whether over-age
  enrolment differs by sex; if it does, the gap is between 0 and 11 points
  and boys are the disadvantaged group. Not reported as a finding.

Referral completion by disability status, 1,638 consenting cases

  Disability reported     26.7%   n = 202
  Not reported            46.2%   n = 1,436
  Difference -19.4 points, 95% CI -26.1 to -12.8, p < 0.001

  The smallest gap consistent with the data is 12.8 points. Reported as a
  finding, and the pathway lesson locates it at referral-making and
  acceptance rather than at consent.
```

Both blocks report the difference and its interval as the headline, with the two group figures beneath. That ordering is deliberate: the difference is the claim, and the two proportions are the evidence for it.

3.8 What comes next

One of these gaps is statistically significant. The next lesson asks whether that is the same thing as mattering — and finds a comparison where a p-value below 0.001 describes a difference no programme would act on.

CHAPTER 4

p = 0.023, and a third of one day

Lesson 4 of 8 · 180 min

Boys attend 88.77% and girls 88.21%. On 68,267 attendance marks that is significant at p = 0.023. It is also a difference of one third of one school day per child per term, which no programme would act on and none should.

4.1 A significant result nobody should act on

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

attendance = pd.read_csv("school-attendance-2024.v1.csv")
enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
current = enrolment[enrolment["school_year"] == 2024]

MARKS = {"true": True, "Y": True, "false": False, "N": False}
marked = attendance[attendance["present"].isin(MARKS)].copy()
marked["attended"] = marked["present"].map(MARKS)
marked = marked.merge(current[["student_id", "sex"]], on="student_id")

by_sex = marked.groupby("sex")["attended"].agg(["sum", "size"])
print(by_sex)

stat, p = proportions_ztest(by_sex["sum"], by_sex["size"])
print(f"z = {stat:.2f}, p = {p:.4f}")
```

```r
library(dplyr)

marked |> summarise(k = sum(attended), n = n(), .by = sex)
prop.test(c(29655, 30749), c(33408, 34859))
```

Group	Attended	Marks	Rate
Boys	29,655	33,408	**88.77%**
Girls	30,749	34,859	**88.21%**

Difference 0.56 percentage points. z = 2.28, p = 0.023.

By the convention every report uses, that is a significant difference in attendance by sex. It would pass a reviewer. It is also, operationally, nothing.

4.2 Convert it into the unit of the decision

```python
term_days = marked["attendance_date"].nunique()
gap = 0.0056
print(f"{term_days} school days in the term")
print(f"0.56 points = {gap * term_days:.2f} days per student per term")
```

```
# The statistic is in percentage points. The decision is in days.
```

Sixty school days, so 0.56% is 0.34 days — about one third of one day per child per term.

No attendance intervention is designed at that resolution. No head teacher can act on it. No budget line moves. **The result is real, replicable, statistically significant and operationally empty**, and those four things are entirely compatible.

Always convert an effect into the unit the decision is made in. Percentage points are the analyst's unit; days, children, cases and dollars are the programme's.

4.3 Why p got small: it was the n

The p-value answers "how surprising would this data be if there were no difference at all", and surprise grows with sample size for any fixed effect.

```
for scale in (0.05, 0.2, 1.0):
    k = (by_sex["sum"] * scale).round().astype(int)
    n = (by_sex["size"] * scale).round().astype(int)
    stat, p = proportions_ztest(k, n)
    print(f"n = {n.sum():>6,}  difference {k[1]/n[1] - k[0]/n[0]:+.2%}  p = {p:.3f}")
```

```
# Same proportions, smaller n, and the p-value walks back across 0.05.
```

Sample	Difference	p
5% of the marks (n=3,413)	+0.62%	0.570
20% (n=13,654)	+0.55%	0.314
100% (n=68,267)	**+0.56%**	**0.023**

The effect never changed. Only the sample did. A p-value is a statement about evidence, not about magnitude, and with a large enough sample every difference becomes significant.

So a p-value alone can never tell you whether something matters. It tells you whether you can rule out zero, and zero is rarely the interesting comparison.

4.4 Report an effect size beside every p-value

Three effect measures cover almost everything a programme report needs.

Measure	For	Read as
Difference in percentage points	Two proportions	The direct answer
Risk ratio	Two proportions, rare outcomes	How many times more likely
Cohen's d	Two means	Standard deviations of separation

```
p_boys, p_girls = by_sex["sum"] / by_sex["size"]
print(f"difference: {p_boys - p_girls:+.2%} points")
print(f"risk ratio: {p_boys / p_girls:.3f}")
```

```
# Two lines. There is no excuse for a p-value with no effect size beside it.
```

Risk ratio 1.006. Boys attend 0.6% more often in relative terms — a way of saying the same nothing.

Contrast that with the disability gap from the last lesson: a difference of 19.4 points and a risk ratio of 0.58, meaning cases reporting a disability complete at just over half the rate. **The same two numbers, computed the same way, describe a finding in one case and noise in the other**, and only the effect size distinguishes them.

4.5 Set the threshold before you test

The defence against both errors — chasing a significant nothing, and dismissing a non-significant something — is to decide in advance what size of difference would change what you do.

```
MEANINGFUL = 0.03      # 3 points of attendance, about 1.8 days a term

observed = p_boys - p_girls
print(f"observed {observed:+.2%}, threshold {MEANINGFUL:.0%}")
print(f"large enough to act on: {abs(observed) >= MEANINGFUL}")
```

```
# Write the threshold in the analysis plan, not after seeing the result.
```

Three points of attendance is about 1.8 days a term, which is the scale at which a follow-up programme would be designed. The observed 0.56 is a fifth of that.

Writing `MEANINGFUL` down before running the test is what stops the threshold moving to wherever the result landed. It is the same discipline as the confounding table in the epidemiology course: **decide the rule before you see the number it will be applied to.**

4.6 The four cases

```
cases = pd.DataFrame([
    ("significant, large",     "report it",                    "disability gap, -19.4 pts"
    ),
    ("significant, small",     "report the effect size, say it is small", "attendance by
    sex, 0.56 pts"),
    ("not significant, large", "report the interval; underpowered", "over-age by sex, +5.5
     pts"),
    ("not significant, small", "report as no evidence of a difference", "-"),
], columns=["case", "what to do", "example in this course"])
print(cases)
```

```
# Four quadrants, and only one of them is "publish the p-value".
```

The third row is the one that gets mishandled. The over-age sex gap was 5.5 points with p = 0.066, and the instinct is to report "no difference". The correct report is that the study could not settle a difference of a size that would have mattered — which is a statement about the sample, and an argument for a larger one rather than for closing the question.

4.7 Report it whole

```
Attendance by sex, February to April 2024

  Boys    88.77%   29,655 of 33,408 marks
  Girls   88.21%   30,749 of 34,859 marks

  Difference +0.56 percentage points (p = 0.023, risk ratio 1.006).

  Equivalent to 0.34 school days per child per term. Below the 3-point
  threshold set in the analysis plan and not reported as a programme
  finding. The p-value is small because the analysis has 68,267 marks, not
  because the difference is large.
```

The last sentence is what makes the block honest. A reader who sees only "p = 0.023" will assume the difference matters, and the sentence that says why it does not is one line long.

4.8 What comes next

Every test so far has been one comparison. The next lesson runs twenty-three at once, on an effect that does not exist, and counts how many come back "significant".

Part III

Where comparisons break

CHAPTER 5

Twenty-four tests, two findings, no effect

Lesson 5 of 8 · 180 min

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.

5.1 A comparison with a known answer

The last two lessons tested one comparison at a time. A real analysis rarely stops at one: a reviewer asks for the same indicator by district, by grade, by school, by displacement status, and the analyst runs each one.

This lesson runs twenty-four of them on a comparison whose true answer is known, because the dataset was generated with sex drawn independently of age, grade, repetition and school. **The true over-age difference by sex is exactly zero**, and anything a test finds is an artefact of sampling.

```python
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"])
    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")
```

```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)
```

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

Twenty-four schools tested, two significant at p < 0.05. Both would be written up. SCH23 in particular looks striking — boys over-age at more than twice the rate of girls, in a school of 43 children.

Neither is real. The generator never let sex touch anything.

5.2 Two is what chance owed you

```
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%}")
```

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

Expected false positives: 1.2. Observed: 2. The probability of getting *at least one* significant result from twenty-four true nulls is **70.8%** — so the surprising outcome would have been finding none.

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 time in twenty. Run twenty tests and being misled once is the expected case, not the accident.

This is why "we looked at it by school and two schools stood out" is not a finding. It is a description of what twenty-four tests do.

5.3 Bonferroni: divide the threshold by the number of tests

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

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

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

The correction is crude — it assumes the worst case, that every test is independent and every null is true — and it is crude in the safe direction. **It is the right default when you do not have a reason to prefer something else**, and it takes one line.

5.4 Benjamini–Hochberg, when twenty-four becomes two hundred

Bonferroni controls the chance of *any* false positive, which is strict. When a screening analysis runs hundreds of tests and expects some real effects among them, controlling the *proportion* of false discoveries is the more useful guarantee.

```python
from statsmodels.stats.multitest import multipletests

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

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

Benjamini–Hochberg also rejects none of the twenty-four, which is the correct answer here — there is nothing to find. Its advantage appears when there is: against two hundred tests with twenty real effects, Bonferroni will miss most of the twenty and BH will not.

Use Bonferroni for a small confirmatory family; use BH for a large exploratory screen. Both are one line, and the choice matters far less than making one.

5.5 Correction does not destroy real findings

The fear that stops people correcting is that it will bury something true. Test a real effect the same way and watch what happens.

```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")
```

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

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**

The 94.5% versus 30.6% gap from the education course survives correction in nine of the fourteen schools where it can be tested. A gap that does not exist survives nowhere. The correction is not a tax on findings; it is what separates them from the two schools that were always going to look interesting.

5.6 Count the tests, including the ones you did not report

The count that goes into the correction is the number of tests you *ran*, not the number you *wrote up*. Four kinds of test are routinely uncounted:

Subgroups you looked at and dropped. Splitting by district, seeing nothing, and splitting by grade instead is two families of tests, not one.

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 striking subgroup result.

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

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

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

5.7 Report it whole

```
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.
```

The comparison row is what makes the block persuasive rather than defensive. It shows the same procedure finding something when there is something to find, which is what a programme manager needs to see before accepting that two striking schools are noise.

5.8 What comes next

Every test so far has assumed one row is one independent observation. The next lesson takes a comparison where that assumption fails — a school feeding programme assigned by school, tested on students — and finds a t-statistic that is nearly twice as large as the honest one.

CHAPTER 6

Twelve hundred students, twenty-four decisions

Lesson 6 of 8 · 180 min

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.

6.1 Check who was assigned before you check who was measured

```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())
```

```r
library(dplyr)

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

Not one school of the twenty-four has both fed and unfed students. Fifteen schools run the programme and nine do not, and every child in a school shares its status.

The roster has 1,202 rows for 1,200 students, and that has to be settled first: two children were transferred without being de-registered, so each appears under both schools with opposite feeding status. Joining before de-duplicating gives them two attendance rows apiece and puts the same child on both sides of the comparison.

That single line decides the whole analysis. **The programme was assigned to 24 schools, so the analysis has 24 independent observations, not 1,200** — and the 1,200 students are 24 groups of about fifty children who share a head teacher, a catchment, a road and a term calendar.

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.

6.2 The two analyses

```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()
```

```python
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))
```

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

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

```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}")
```

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

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

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 effect barely moved. The standard error nearly doubled. That is the whole phenomenon: treating clustered observations as independent does not bias the estimate, it understates its uncertainty, and every p-value and interval built on it is too narrow.

6.3 Why the ratio is 1.8

The multiplier is not arbitrary — it comes from how much of the variation sits *between* schools rather than within them.

```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}")
```

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

ICC 0.060, average cluster 50, design effect 3.94. Six per cent of the variation in attendance is between schools, which sounds negligible and is not: with fifty children per school it multiplies the variance nearly fourfold.

Effective sample size 304, not 1,200. The standard error grows by the square root of the design effect, and the square root of 3.94 is 1.99 — which is the 1.8 observed above.

This is the same design effect the survey course applied to cluster sampling. **The arithmetic does not care whether the clustering came from a sampling design or from how a programme was rolled out** — a school feeding programme assigned by school is a cluster design whether anyone called it one.

6.4 Three ways to get it right

Aggregate to the unit of assignment. Compute one number per school, test the

1. Simple, transparent, and it is what the table above does. The cost is that a

   school of 90 children counts the same as a school of 20.

```
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%}
    ")
```

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

Use a mixed model with a random intercept for school. Keeps every student, estimates the between-school variance explicitly, and handles unequal school sizes. `statsmodels.formula.api.mixedlm` in Python, `lme4::lmer` in R.

Use cluster-robust standard errors. Keeps the student-level model and corrects the standard error for clustering, which is the standard approach in econometrics and is one argument in `statsmodels`. It needs a reasonable number of clusters — 24 is on the low side, and below about 30 the correction itself becomes unreliable.

With 24 clusters, aggregate. The mixed model buys precision when there are many clusters of unequal size; here it would produce a similar answer with more machinery and more assumptions.

6.5 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 school level.** Both are correct answers to different questions, and reporting the student-level figure as "the correlation between school feeding and attendance" attributes a school-level relationship to children.

```
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}")
```

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

Water points. Functionality is assigned by point and measured by visit. The WASH course counted a point once and not its twelve visits, which is this rule applied before it had a name.

Referral cases. A case worker with forty cases is one worker, and comparing outcomes across case workers on 1,600 cases has however many workers there are as its unit.

Repeated measures on the same household. Three survey rounds on one household are three rows and one household, and the same correction applies.

6.6 Read the ratio as a claim about generalisation

There is a reason the honest analysis feels weaker, and it is not a technicality.

With 24 schools, "does school feeding raise attendance" is being answered by fifteen schools that have it against nine that do not. Everything that differs between those two sets of schools — where they are, who runs them, why they were chosen for the programme — rides along with the comparison, and 1,200 students do nothing to separate it.

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 children had each been independently assigned to receive school meals, which would have told you far more and would have been a different programme.

6.7 Report it whole

```
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.
```

The last paragraph costs two lines and is the one a reviewer will look for. Getting the unit of analysis right makes the interval honest; it does not make the comparison causal, and the epidemiology course established what does.

6.8 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.

Part IV

What the report claims

CHAPTER 7

r = 0.039, and everyone expected 0.5

Lesson 7 of 8 · 180 min

Attendance and endline literacy correlate at 0.039 across 659 students — an interval from −0.04 to +0.12, which is as close to nothing as a dataset gets. The correlation everyone assumes is in the register is not in this one, and reporting that is the finding.

7.1 The correlation that is not there

```python
import pandas as pd
import numpy as np

assessment = pd.read_csv("learning-assessment-2024.v1.csv")
attendance = pd.read_csv("school-attendance-2024.v1.csv")
roster = pd.read_csv("school-roster-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)
rate = marked.groupby("student_id")["attended"].mean().rename("attendance")

endline = (assessment[assessment["assessment_round"] == "endline"]
           .pivot_table(index="student_id", columns="domain", values="raw_score"))
joined = endline.join(rate, how="inner").join(roster.set_index("student_id"))

print(joined[["literacy", "numeracy", "attendance"]].corr().round(3))
```

```r
library(dplyr)

joined |> select(literacy, numeracy, attendance) |> cor(use = "complete.obs")
```

Pair	r	n
Attendance and endline literacy	**0.039**	659
Attendance and endline numeracy	**−0.006**	659
Literacy and numeracy	0.849	659

Attendance and literacy correlate at 0.039. Every education programme document assumes that relationship and most quote a number for it. In this register it is absent.

That is a result, and it needs an interval before it is one.

7.2 An interval on a correlation

```python
def fisher_ci(r, n, z=1.96):
    zr = np.arctanh(r)
    se = 1 / np.sqrt(n - 3)
    return np.tanh(zr - z * se), np.tanh(zr + z * se)
```

```python
for label, r, n in [("attendance vs literacy", 0.039, 659),
                    ("literacy vs numeracy", 0.849, 659)]:
    lo, hi = fisher_ci(r, n)
    print(f"{label:24} r = {r:+.3f}  95% CI [{lo:+.3f}, {hi:+.3f}]")
```

```r
cor.test(joined$attendance, joined$literacy)
```

r = 0.039, 95% CI −0.037 to +0.115. The interval contains zero and every value inside it is negligible, so this is not an underpowered null — 659 students are enough to say that if a relationship exists it is too small to matter.

Contrast the same interval on the second pair: **r = 0.849, CI 0.826 to 0.869**, where the whole interval is large.

A correlation with no interval is the same defect as a proportion with none, and `cor.test` gives it for free.

7.3 The correlation that is there and says nothing

Literacy and numeracy correlate at 0.849, which is the largest r in this course and the least interesting.

It is one child, one test day, two forty-item papers. A child who was ill, who had not eaten, who could not read the instructions, or who is simply a strong student scores similarly on both. The correlation is a property of the measurement, not a discovery about literacy and numeracy.

Ask what would have to be true for the correlation to be zero. Here the answer is "the same child would have to perform independently on two papers taken half an hour apart", which nobody believes — so a high r carries no information.

A correlation is only informative when its absence was plausible, and that is a judgement about the world rather than about the data.

7.4 Square it before you describe it

```python
for label, r in [("attendance vs literacy", 0.039),
                 ("attendance vs literacy, school level", 0.199),
                 ("literacy vs numeracy", 0.849)]:
    print(f"{label:38} r = {r:+.3f}   r-squared = {r**2:.3f}")
```

```r
# r^2 is the share of variance, and it is much smaller than r looks.
```

Pair	r	r^2	Reads as
Attendance and literacy	0.039	**0.002**	Two tenths of one per cent
Attendance and literacy, school level	0.199	0.040	Four per cent
Literacy and numeracy	0.849	0.721	Seventy-two per cent

An r of 0.199 sounds like a relationship and explains four per cent of the variation. Squaring is the cheapest defence against over-reading a middling correlation, and the number to put in a report is usually r^2 rather than r.

7.5 Pearson, Spearman, and eleven bad rows

Lesson 1 found eleven households whose *total* consumption was entered in a per-person column. Watch what those eleven rows do to a correlation.

```python
wash = pd.read_csv("wash-household-survey-2024.v1.csv")
pair = wash.dropna(subset=["round_trip_minutes", "litres_per_person_day"])
clean = pair[pair["litres_per_person_day"] <= 80]

for label, frame in [("all rows", pair), ("11 rows removed", clean)]:
    p = frame["round_trip_minutes"].corr(frame["litres_per_person_day"])
    s = frame["round_trip_minutes"].corr(frame["litres_per_person_day"],
                                         method="spearman")
    print(f"{label:18} n = {len(frame):5}  Pearson {p:+.3f}  Spearman {s:+.3f}")
```

```r
cor(pair$round_trip_minutes, pair$litres_per_person_day)                    # Pearson
cor(pair$round_trip_minutes, pair$litres_per_person_day, method = "spearman")
```

Rows	n	Pearson	Spearman
All	2,403	**−0.185**	−0.285
Eleven removed	2,392	**−0.293**	−0.287

Eleven rows in 2,403 cut Pearson by more than a third. Spearman did not move.

Pearson uses the values, so a household recorded at 277 litres per person pulls hard on the coefficient. Spearman uses the ranks, so a wrong value that is still the largest value changes nothing.

Report Spearman when either variable is skewed or has outliers you have not resolved, and Pearson when both are roughly symmetric and clean. A large gap between the two is a signal to go and look at the extremes — the same use the skewness statistic had in lesson 1.

The corrected answer is the useful one: longer trips to water go with less water used, r = −0.29 on 2,392 households, which is the relationship the WASH course predicted and the uncorrected file understated.

7.6 Correlated because something else moved both

```python
prices = pd.read_csv("market-prices-2024.v1.csv")
wide = prices.pivot_table(index="period", columns="commodity",
                          values="price_htg", aggfunc="mean").sort_index()

print(f"beans and maize, levels:           {wide['beans-black'].corr(wide['maize']):.3f}"
    )
changes = wide.diff().dropna()
print(f"beans and maize, month-on-month:   {changes['beans-black'].corr(changes['maize'])
    :.3f}")
```

```r
cor(wide$`beans-black`, wide$maize)
cor(diff(wide$`beans-black`), diff(wide$maize))
```

Beans and maize correlate at 0.986 in levels and 0.876 in month-on-month changes. Neither number means the price of beans moves the price of maize. Both series carry the same annual inflation and the same lean-season shock, and the correlation is measuring the shock.

Correlating two series over time will nearly always give a large number, because almost everything trends. Differencing removes the shared trend and what survives is the shared shock — which is a common cause, and still not a causal link between the two commodities.

The epidemiology course made this point with confounding. **The correlation coefficient has no way to express it**, which is why the sentence beneath it has to.

7.7 And the clustering from the last lesson

```
by_school = joined.groupby("school_id")[["attendance", "literacy"]].mean()
print(f"student level  r = {joined['attendance'].corr(joined['literacy']):.3f}  n = {len(
    joined)}")
print(f"school level   r = {by_school['attendance'].corr(by_school['literacy']):.3f}  n =
    {len(by_school)}")
```

```
# Same two variables, two units of analysis, two answers.
```

0.039 across 659 students and 0.199 across 24 schools. The school-level figure has an interval from −0.22 to +0.56 on 24 points, so it establishes nothing either — but it is five times larger, and an analyst who aggregated first and reported r without n would have written a different sentence.

Say which unit the correlation is between, every time. "Attendance correlates with literacy at 0.20" is unreadable without knowing whether the rows are children or schools.

7.8 Write the sentence

The coefficient is one number and the sentence beneath it is where the analysis is either honest or not.

Not this: "Attendance is correlated with literacy outcomes." True of the school level, false of the student level, and unfalsifiable as written.

Nor this: "There is no relationship between attendance and literacy." Stronger than the data supports; the interval reaches +0.12.

This: "Across 659 students, attendance over the February–April term shows no association with endline literacy (r = 0.04, 95% CI −0.04 to +0.12). The register cannot support the assumption that attendance drives learning in this cohort. Note that attendance varies little — the middle half of students sit between 86.7% and 97.8% — so this analysis has limited ability to detect a relationship at the low attendance where one would be expected."

The third sentence is the one to copy. It states the finding, the interval, what it means for the programme, and the reason the analysis might have missed something real — and it is four lines.

7.9 Report it whole

```
Attendance and learning outcomes, endline 2024
```

```
Attendance vs endline literacy   r = 0.04   95% CI -0.04 to +0.12   n = 659
Attendance vs endline numeracy   r = -0.01  95% CI -0.08 to +0.07   n = 659

Between students within the term. No association at student level.
Aggregated to the 24 schools, r = 0.20 (95% CI -0.22 to +0.56), which is
also consistent with no relationship.

Attendance is compressed: the interquartile range is 86.7% to 97.8%, so
students with the low attendance that a learning effect would show up at
are rare in this cohort. This is a null result about the range observed,
not about attendance in general.
```

The last paragraph is what stops the null being over-read, and it is the counterpart of the "underpowered, not null" sentence from lesson 4. A correlation of zero over a narrow range says nothing about what happens outside it.

7.10 What comes next

The last lesson assembles all of this into the statistics section of a report — what goes in, what stays in the appendix, and what a reviewer will ask for that this course has already computed.

CHAPTER 8

The statistics section a reviewer cannot dismantle

Lesson 8 of 8 · 180 min

Seven lessons produced eleven numbers. This one puts them into a report — what goes in the body, what goes in the annex, what a reviewer will ask for, and the four sentences that answer every question before it is asked.

8.1 What a reviewer actually does

A reviewer reading a programme report does four things, in this order, and each one is answered by something this course computed.

The reviewer asks	Answered by
"How sure are you of this number?"	The interval — lesson 2
"Is that difference real?"	The test and its assumptions — lesson 3
"Does it matter?"	The effect size in the decision's unit — lesson 4
"How many things did you look at?"	The test count and the correction — lesson 5

The order is not the order you computed them in. The interval comes first because it is the only one a reader can check against their own sense of the programme, and the test count comes last because it is the one they will forget to ask about.

8.2 The four sentences

Every statistical claim in a report can be written in four sentences, and the structure below is what the "report it whole" block at the end of each lesson has been rehearsing.

One: the number and its interval. "Referral completion among cases reporting a disability is 26.7% (95% CI 21.1 to 33.2, n = 202)."

Two: the comparison and its test. "That is 19.4 points below cases with no disability reported (46.2%, n = 1,436; 95% CI on the difference −26.1 to −12.8, two-sample z-test, $p < 0.001$)."

Three: the size, in the unit of the decision. "Applied to the 2024 caseload, the gap is 39 cases that would have reached a service had completion matched the rest of the caseload."

Four: what the analysis cannot say. "The comparison is observational. Cases are not randomised and the gap may reflect where cases reporting a disability enter the system rather than how they are handled once inside."

```
import pandas as pd
import numpy as np

protection = pd.read_csv("protection-referrals-2024.v1.csv")
consenting = protection[protection["consent_to_refer"]]
disability = consenting["disability_reported"].isin([True, "true", "Yes"])
reached = consenting["referral_accepted"] & consenting["days_to_first_service"].notna()

k, n = reached[disability].sum(), disability.sum()
comparator = reached[~disability].mean()
```

```
print(f"cases short of the comparator rate: {comparator * n - k:.0f}")


# Sentence three is arithmetic, and it is the sentence a manager reads.
```

Sentence three is the one analysts skip and managers need. Percentage points do not commission anything; thirty-nine cases do.

8.3 Body, annex, and script

Three places, and putting a number in the wrong one is the commonest failure of a statistics section.

The body carries the claim. The estimate, its interval, the comparison, the effect in programme units, and the limitation. Five lines per finding, no more.

The annex carries the machinery. Test names, statistics, degrees of freedom, assumption checks, the count of comparisons run, the correction applied, and the subgroup analyses that found nothing.

The script carries the reproduction. Every number in both, generated by code that runs from the committed file — which is why every "report it whole" block in this course is printed by the same script that computed it.

```
def claim(label, k, n, digits=1):
    p = k / n
    z = 1.96
    denom = 1 + z**2 / n
    centre = (p + z**2 / (2 * n)) / denom
    half = z * np.sqrt(p * (1 - p) / n + z**2 / (4 * n**2)) / denom
    return (f"{label}: {p:.{digits}%} "
            f"(95% CI {centre - half:.{digits}%} to {centre + half:.{digits}%}, n={n})")

print(claim("Referral completion, disability reported", 54, 202))


# Generate the sentence, do not type it. A typed interval drifts from its number.
```

A number typed into a document is a number that will be wrong after the next data correction. Generating the sentence is the same discipline as generating the figures from the dataset.

8.4 The annex table

```
annex = pd.DataFrame([
    {"comparison": "Completion by disability status", "test": "two-sample z",
     "statistic": "z = -5.21", "p": "<0.001", "n": "202 vs 1,436",
     "effect": "-19.4 pts (CI -26.1 to -12.8)"},
    {"comparison": "Over-age by sex", "test": "two-sample z",
     "statistic": "z = 1.84", "p": "0.066", "n": "510 vs 542",
     "effect": "+5.5 pts (CI -0.4 to +11.3)"},
    {"comparison": "Attendance by sex", "test": "two-sample z",
     "statistic": "z = 2.28", "p": "0.023", "n": "68,267 marks",
     "effect": "+0.56 pts, risk ratio 1.006"},
    {"comparison": "Attendance by school feeding", "test": "Welch t, school level",
     "statistic": "t = 3.11, df 12.7", "p": "0.008", "n": "15 vs 9 schools",
     "effect": "+5.2 pts (CI 1.6 to 8.8)"},
```

```
])
print(annex.to_string(index=False))


# One row per test run, including the ones that found nothing.
```

Comparison	Test	Statistic	p	Effect
Completion by disability	Two-sample z	z = −5.21	<0.001	−19.4 pts
Over-age by sex	Two-sample z	z = 1.84	0.066	+5.5 pts
Attendance by sex	Two-sample z	z = 2.28	0.023	+0.56 pts, RR 1.006
Attendance by feeding	Welch t, school	t = 3.11, df 12.7	0.008	+5.2 pts
Over-age by sex, 24 schools	Two-sample z ×24	2 of 24 at p<0.05	—	0 survive Bonferroni

Three of those five rows are in the annex and not in the body, because a non-significant gap, a significant irrelevance and a family of null tests are all things a reviewer needs to be able to find and none of them is a programme finding.

Listing the tests that found nothing is what makes the ones that found something credible. An annex containing only successful tests tells a reviewer that the count was managed.

8.5 The limitations paragraph, written from the course

Four limitations recur across every analysis in this platform, and each has a one-line form that is honest rather than defensive.

Observational comparison. "Groups were not randomised; differences may reflect who enters each group rather than what happens to them."

Clustering. "The programme is assigned by school, so the analysis uses 24 independent units. Student-level statistics would overstate precision by a factor of 1.8."

Multiple comparisons. "Twenty-four school-level tests were run; 1.2 significant results are expected by chance and 2 were observed, neither surviving correction."

Range restriction. "Attendance in this cohort runs from 86.7% to 97.8% in the middle half, so this analysis cannot speak to the low-attendance students where an effect would be expected."

Each is one sentence and each pre-empts a specific question. A limitations paragraph that says "the data have limitations" pre-empts nothing and reads as having something to hide.

8.6 What to do when the number will not settle

Three of this course's results are genuinely unresolved, and each has a different correct action.

Result	Why it is unresolved	What to do
Over-age by sex, p = 0.066	Underpowered for a 5-point gap	Say so; pool the next round rather than clos
Attendance and literacy, r = 0.04	Range restriction	Report the null and name the restriction
Feeding and attendance, 24 clusters	Few clusters, observational	Report the interval; do not claim an effect

"The data cannot settle this" is a finding a programme can act on, because the action is to collect differently. It is not the same as having nothing to report, and the version that gets a programme into trouble is the one that reports a number anyway.

8.7 Report it whole

```
Statistical methods

  Proportions are reported with Wilson 95% confidence intervals. Group
  comparisons use two-sample z-tests for proportions and Welch's t-test for
  means. Where a programme is assigned above the level of the observation,
  the analysis is conducted at the level of assignment.

  Analyses were specified before the data were examined. Where a family of
  comparisons was run, the number of tests is stated with the result and a
  Bonferroni correction applied.

  All figures are generated from the committed dataset files by the scripts
  in this annex; no number in this report is typed by hand.

  Findings reported in the body: 2. Comparisons run in total: 29.
```

The last line is the one that changes how the section is read. Two findings from twenty-nine comparisons is a defensible ratio stated openly; two findings with no denominator is a claim a reviewer has no way to weigh.

8.8 Where this goes next

Module 5's remaining two courses take this further. **Regression** replaces the one-comparison-at-a-time discipline of this course with a model that adjusts for several things at once — and inherits every problem here, including the clustering and the multiple comparisons. **Impact evaluation** takes on the fourth sentence directly: which designs let you write *this caused that* instead of *these two groups differ*.

But the discipline is complete here. Every number in a programme report can carry an interval, an effect size, a test count and a limitation, and doing so costs four sentences. **The courses that follow make the numbers better; this one makes them honest**, and no amount of the former substitutes for the latter.

About this edition

This PDF is generated from the same source as the web version of the course, so the two cannot drift. The web edition carries the runnable code, the downloadable datasets and any corrections issued since this file was produced.

> Every dataset referenced in this course is synthetic or fully de-identified. No record traces to a real person.

This course is published in English and in French. The French edition is a professional adaptation using the terminology UNICEF, WHO, WFP, OCHA and national ministries publish in — not a literal translation.

Read and run the course online at data-analysis.cassion.dev/courses/applied-statistics-for-programmes

Generated July 28, 2026.