CASSION · COURSE

# Education Programme Analysis

Enrolment, attendance, retention and learning outcomes, with the school-year calendar and the enrolment-versus-attendance distinction that trips up every dashboard.

| | |
|---|---|
| Instructor | Pierre Robentz Cassion |
| Level | Intermediate |
| Learner hours | 14 h |
| Taught in | Python · R |
| Updated | July 28, 2026 |
| Sectors | Education |
| Standards and methodologies | Sustainable Development Goals (SDG) · UNICEF indicator definitions · Multiple Indicator Cluster Survey (MICS) |

Read and run the course online at data-analysis.cassion.dev/courses/education-programme-analysis

# What you will be able to do

- Compute gross and net enrolment ratios and explain why one of them exceeds 100% without any child being counted twice
- Separate average daily attendance from the proportion of students regularly attending, and say which a programme is judged on
- Follow a cohort through promotion, repetition and dropout, and compute survival to the final grade
- Read a learning assessment across two rounds whose instruments differ, and say what is comparable
- Detect a non-random assessment subsample and bound the bias it introduces
- Disaggregate by sex, disability and displacement, and state the sample each cut demands

## Prerequisites

None. This course assumes no prior programming experience.

# Contents

# About this course

Education is the sector where the most-quoted indicator is the one most often computed against the wrong denominator, and where the distinction a programme lives or dies by — enrolled against attending against learning — is collapsed on almost every dashboard into a single number called *coverage*.

The course runs on four files. The **attendance register** and its roster have appeared in three earlier courses as a cleaning and joining problem; here they are analysed as attendance. Three are new: a **two-year enrolment register** with age, sex, disability, displacement and an end-of-year status; the **school-age population** that is its denominator; and a **two-round learning assessment**.

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

**Gross enrolment is 109.2% and net enrolment is 97.4%.** Neither is wrong and neither is the other. 36.4% of primary enrolment is above the official age for its grade and one child in nine is twelve or older while still in primary — which is what puts the gross ratio over 100% without a single child being counted twice.

**Average daily attendance is 88.4% and 62.1% of students attend at least 90% of the days they were marked.** A programme reporting the first is describing a system; a programme acting on the second is describing children.

**Dropout is 19.5% among over-age students against 4.8% among the rest.** Being behind is the strongest predictor in this register, and it compounds: 94.5% of the children who repeated a grade in 2023 are over-age in 2024, and an over-age child is four times as likely to leave.

Both Python and R throughout. You should have done module 3 — this course assumes you can defend a denominator, read a censored register and put an interval on a proportion.

# Part I

# Enrolment and its denominator

CHAPTER 1

# 109% enrolled, 97% enrolled

Lesson 1 of 8 · 105 min

*Two ratios from one register. The gross is 109.2% and the net is 97.4%, both are correct, and the twelve points between them are children who are in school at the wrong age rather than children who are missing.*

## 1.1 Two ratios, one register

```python
import pandas as pd

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
population = pd.read_csv("school-age-population-2024.v1.csv")

current = enrolment[enrolment["school_year"] == 2024]
primary = current[current["grade"].between(1, 6)]

denominator = population.loc[
    (population["reference_year"] == 2024)
    & population["age_years"].between(6, 11),
    "projected_population",
].sum()

gross = len(primary) / denominator
net = len(primary[primary["age_years"].between(6, 11)]) / denominator

print(f"gross enrolment ratio: {gross:.1%}")
print(f"net enrolment ratio:   {net:.1%}")
```

```r
library(dplyr)

primary <- enrolment |> filter(school_year == 2024, between(grade, 1, 6))
denominator <- population |>
  filter(reference_year == 2024, between(age_years, 6, 11)) |>
  summarise(n = sum(projected_population)) |> pull(n)

tibble(
  gross = nrow(primary) / denominator,
  net = sum(between(primary$age_years, 6, 11)) / denominator
)
```

| Ratio | Numerator | Denominator | Value |
|---|---|---|---|
| **Gross** | All primary enrolees, any age | Children aged 6–11 | **109.2%** |
| **Net** | Primary enrolees aged 6–11 | Children aged 6–11 | **97.4%** |

**The numerators differ; the denominator is identical.** That is the whole definition, and it is why the two ratios are not two estimates of one thing.

## 1.2 Why a ratio can exceed 100%

A gross ratio above 100% is not an error. Three things produce it and only one is a defect.

**Over-age and under-age enrolment.** Children outside the official age range are in the numerator and not in the denominator. This is the dominant cause here and it is a real feature of the system, not a data problem.

**A denominator that is too small.** The population file is a projection, not a count, and a projection that undershoots inflates every ratio built on it.

**Double counting.** A child enrolled at two schools appears twice. This one *is* a defect, and this register has it.

```
duplicates = current.duplicated(subset=["student_id", "school_year"], keep=False)
print(f"student-years appearing more than once: {duplicates.sum()}")

unique = current.drop_duplicates(subset=["student_id", "school_year"])
unique_primary = unique[unique["grade"].between(1, 6)]
print(f"gross ratio after deduplication: {len(unique_primary) / denominator:.1%}")
```

```
enrolment |> filter(school_year == 2024) |>
  count(student_id) |> filter(n > 1) |> nrow()
```

Twelve students appear twice, once under each school, because a transfer was recorded as a new enrolment rather than a move. **Deduplicate on student and year before either ratio**, and note that the effect here is small — the point is not the size, it is that a register which double-counts is wrong about who exists.

## 1.3 The denominator is a projection

```
CENSUS_YEAR, GROWTH = 2015, 0.021
factor = (1 + GROWTH) ** (2024 - CENSUS_YEAR)
print(f"nine years compounded at {GROWTH:.1%}: factor {factor:.3f}")
print(f"so about {1 - 1 / factor:.0%} of the denominator is an assumption")
```

```
(1 + 0.021)^(2024 - 2015)
```

**A factor of 1.21, so roughly a sixth of the denominator was never counted.** This is the immunisation denominator from the public health course, in a different sector: a census base, a growth assumption, and nine years of compounding.

The consequence is specific. **If the projection is 5% too low, the gross ratio falls from 109.2% to about 104%** and the net from 97.4% to about 93%. Neither conclusion changes, but a report claiming primary enrolment rose two points between years may be reporting the growth rate rather than the schools.

```
for error in (-0.05, 0.0, 0.05):
    adjusted = denominator * (1 + error)
    print(f"projection {error:+.0%}: gross {len(primary) / adjusted:.1%}, "
          f"net {len(primary[primary['age_years'].between(6, 11)]) / adjusted:.1%}")
```

```
# Vary the denominator and see which conclusions survive.
```

**Show the sensitivity rather than the point estimate** where the denominator is projected. It costs three lines and it is the difference between a ratio and a ratio you can defend.

## 1.4 Which one to report

Neither, alone.

**Net enrolment answers "are children of school age in school".** It is the SDG 4.1 framing and the right ratio for a coverage question. It cannot exceed 100%, which makes it the safer number to publish.

**Gross enrolment answers "how much primary schooling is being delivered".** It is the right ratio for a capacity question — teachers, classrooms, textbooks — because an over-age child needs a desk exactly as much as an in-age one.

**The gap between them is the finding**, and it is the reason to report both: twelve points of over-age enrolment is a statement about repetition and late entry that neither ratio makes on its own.

```
Primary enrolment, 2024, four districts

  Gross enrolment ratio       109.2%   1,052 enrolees / 963 children aged 6-11
  Net enrolment ratio          97.4%     938 in-age enrolees / same denominator
  Over-age share of enrolment  36.4%   above the official age for their grade

  Denominator is a projection from the 2015 census at 2.1% a year. A 5%
  error in it moves the gross ratio by about five points and changes no
  conclusion.
  12 student-years were recorded twice after transfers and are deduplicated.
```

## 1.5 What the ratios cannot tell you

**Neither is attendance.** A child enrolled and never present is in both numerators. The next unit is that distinction, and it is worth more than either ratio.

**Neither is completion.** Enrolment is a stock at a point in the year; whether those children finish is a cohort question and the third unit.

**Neither is learning.** A system can enrol every child of school age and teach none of them, and the last unit is the instrument that would notice.

## 1.6 What comes next

Twelve points of the gap between the two ratios is over-age enrolment. The next lesson is who those children are, why being behind is the strongest predictor in this register, and how repetition makes it compound.

CHAPTER 2

# Behind, and getting further behind

Lesson 2 of 8 · 105 min

*Over-age enrolment runs 17.0% in grade 1 and 50.8% in grade 6. It rises because repetition creates it — 94.5% of the children who repeated in 2023 are over-age in 2024 — and because being over-age quadruples the chance of leaving.*

## 2.1 The definition, and the range check that has to come first

A student is over-age for their grade when their age exceeds the official age, which in this system is the grade number plus five.

```python
import pandas as pd

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
print(enrolment["age_years"].describe().round(1))
print(f"ages above 20: {(enrolment['age_years'] > 20).sum()}")
```

```r
library(dplyr)
enrolment |> summarise(min = min(age_years), max = max(age_years),
                       impossible = sum(age_years > 20))
```

**Twenty-one rows hold an age above 20 in a primary register.** SCH04 records part of its intake in months, so a nine-year-old appears as 112. A mean would absorb them; a range check finds them, and the correction is a division rather than a deletion.

```python
clean = enrolment[enrolment["age_years"] <= 20].copy()
clean["official_age"] = clean["grade"] + 5
clean["over_age"] = clean["age_years"] > clean["official_age"]
```

```r
enrolment |> filter(age_years <= 20) |>
  mutate(official_age = grade + 5, over_age = age_years > official_age)
```

**Do the range check before the flag, not after.** An age of 112 is over-age for every grade, so a flag computed on the raw column is technically correct about twenty-one children and wrong about what it means.

## 2.2 It compounds with grade

```python
primary = clean[(clean["school_year"] == 2024) & clean["grade"].between(1, 6)]
by_grade = primary.groupby("grade").agg(
    students=("student_id", "size"),
    over_age=("over_age", "mean"),
)
print((by_grade * [1, 100]).round(1))
```

```
primary |> summarise(n = n(), over_age = mean(over_age), .by = grade)
```

| Grade | Official age | Students | Over-age |
|---|---|---|---|
| 1 | 6 | 53 | **17.0%** |
| 2 | 7 | 153 | 25.5% |
| 3 | 8 | 247 | 29.6% |
| 4 | 9 | 258 | 36.8% |
| 5 | 10 | 209 | 47.8% |
| 6 | 11 | 132 | **50.8%** |

**Over-age triples between grade 1 and grade 6.** Two mechanisms produce that shape and they are not the same problem.

**Late entry** puts a child above age at grade 1 and keeps them there. It shows up as the 17.0% floor, and the intervention is early registration.

**Repetition** creates over-age *during* schooling. It shows as the rise, and the intervention is entirely different.

```
previous = clean[clean["school_year"] == 2023].set_index("student_id")
current = clean[clean["school_year"] == 2024].copy()
current["last_year"] = current["student_id"].map(previous["end_of_year_status"])

print(current.groupby("last_year")["over_age"].agg(["mean", "size"]).round(3))
```

```
enrolment |>
  filter(age_years <= 20) |>
  select(student_id, school_year, grade, age_years, end_of_year_status) |>
  tidyr::pivot_wider(names_from = school_year,
                     values_from = c(grade, age_years, end_of_year_status))
```

**94.5% of the students who repeated in 2023 are over-age in 2024, against 30.6% of those who were promoted.** Repetition does not correlate with over-age; it *causes* it, with a one-year lag, and the register lets you watch the mechanism rather than infer it.

Note that the comparison has to be across years. Within 2023 a repeater is not yet over-age — they are in their own grade at the normal age, and the extra year appears the following September.

## 2.3 Why it matters: the dropout link

```
year23 = clean[clean["school_year"] == 2023]
dropout = year23.groupby("over_age")["end_of_year_status"].apply(
    lambda s: (s == "dropped-out").mean()
)
counts = year23.groupby("over_age").size()
print(pd.DataFrame({"dropout": (dropout * 100).round(1), "n": counts}))
```

```
enrolment |> filter(school_year == 2023) |>
  summarise(dropout = mean(end_of_year_status == "dropped-out"),
            n = n(), .by = over_age)
```

| | Students | Dropped out |
|---|---|---|
| **Over-age** | 478 | **19.5%** |
| In-age | 814 | 4.8% |

**Being behind quadruples the chance of leaving**, and that is what closes the loop: a child repeats, becomes over-age, and is then far likelier to drop out than the repetition was intended to prevent.

**So repetition is not a neutral intervention.** It is offered as a second chance and it measurably raises the probability of leaving altogether. That finding is available from two columns of one register, and it is the strongest argument this dataset supports.

## 2.4 The disaggregation that does not work

Over-age is unusual among education indicators in that the obvious cuts show very little.

```python
for column in ("sex", "disability_reported", "displacement_status"):
    print(primary.groupby(column, dropna=False)["over_age"].agg(
        ["mean", "size"]).round(3), "\n")
```

```
primary |> summarise(over_age = mean(over_age), n = n(), .by = sex)
```

| Cut | Over-age | n |
|---|---|---|
| Boys | 39.2% | 510 |
| Girls | 33.8% | 542 |
| Disability reported | 33.7% | 89 |
| No disability reported | 36.7% | 963 |

**Five points between boys and girls, on 510 and 542 students.** That is around the edge of what sampling variation produces, so compute the interval before writing the sentence — the survey course's machinery, on exactly the kind of gap that gets published as a finding and disappears in the next round.

**Disability shows three points the other way on 89 students**, which is noise on a denominator that small.

**A disaggregation that mostly shows nothing is a result.** Over-age here is a system-level phenomenon driven by repetition and late entry, not a group-level inequity — and saying so is more useful than hunting for a subgroup until one turns up.

## 2.5 Report the profile, not the headline

```
Over-age enrolment, primary, 2024

  Overall                    36.4%   383 of 1,052 primary enrolees
  Grade 1                    17.0%   the late-entry floor
  Grade 6                    50.8%   after five years of repetition
  Repeated in 2023           94.5%   over-age in 2024, against 30.6% promoted

  Dropout, over-age          19.5%   against 4.8% in-age (2023 cohort)

  21 ages recorded in months at SCH04, corrected by division.
  Sex difference is 39.2% (boys) against 33.8% (girls); report with an
  interval or not at all.
```

**The grade profile is the deliverable**, because 36.4% overall could mean uniform late entry or accumulating repetition, and those need different programmes.

## 2.6 What comes next

Every child in this lesson is enrolled. Whether they are in the classroom is a different register and a different number — and the next lesson finds two of them, twenty-six points apart, in the same file.

# Part II

# Enrolled is not attending

CHAPTER 3

# 88% attendance, 62% of students

Lesson 3 of 8 · 105 min

*Average daily attendance is 88.4%. The share of students attending at least 90% of their marked days is 62.1%. Both come from the same 70,245 rows, and the twenty-six points between them are the children a mean cannot see.*

## 3.1 One register, two questions

```python
import pandas as pd

attendance = pd.read_csv("school-attendance-2024.v1.csv")
print(f"{len(attendance):,} rows, {attendance['student_id'].nunique():,} students")
print(attendance["present"].value_counts(dropna=False))
```

```r
library(dplyr)
attendance |> count(present)
```

**The `present` column has five values, not two.** SCH09 recorded `Y` and `N` instead of `true` and `false` on 873 of its rows, and 271 days were never marked either way. A boolean cast turns those 873 real observations into missing values and silently thins one school out of the denominator.

```python
MARKS = {"true": True, "Y": True, "false": False, "N": False}
marked = attendance[attendance["present"].isin(MARKS)].copy()
marked["attended"] = marked["present"].map(MARKS)
print(f"usable marks: {len(marked):,} of {len(attendance):,}")
```

```r
attendance |>
  mutate(attended = case_when(present %in% c("true", "Y") ~ TRUE,
                              present %in% c("false", "N") ~ FALSE)) |>
  filter(!is.na(attended))
```

Now the same file answers two different questions.

## 3.2 Average daily attendance

```python
ada = marked["attended"].mean()
print(f"average daily attendance: {ada:.1%}")
```

```r
marked |> summarise(ada = mean(attended))
```

**88.4%.** Every mark counts once, so a student present 60 days out of 60 and one present 30 out of 60 contribute in proportion to how often they were marked.

This is the number a ministry reports and a system-level indicator should be. It answers *how full is the classroom on an average day*, which is the right question for staffing, feeding and textbook planning.

## 3.3 The proportion regularly attending

```python
by_student = marked.groupby("student_id")["attended"].agg(["mean", "size"])
eligible = by_student[by_student["size"] >= 20]

for threshold in (0.80, 0.85, 0.90):
    share = (eligible["mean"] >= threshold).mean()
    print(f"attending at least {threshold:.0%} of marked days: {share:.1%}")
```

```r
marked |>
  summarise(rate = mean(attended), days = n(), .by = student_id) |>
  filter(days >= 20) |>
  summarise(across(everything(), ~ mean(rate >= 0.9)))
```

| Threshold | Students meeting it |
|---|---|
| At least 80% of days | 87.3% |
| At least 85% | 78.9% |
| **At least 90%** | **62.1%** |

**62.1% at the 90% threshold, against an average daily attendance of 88.4%.** The two numbers are twenty-six points apart and neither is wrong.

**A mean over marks describes the system; a proportion over students describes children.** A school where every child misses one day in eight and a school where seven children in eight attend perfectly while one never comes have the same average daily attendance and completely different problems.

## 3.4 Which one a programme is judged on

**Report both, and lead with the one that matches the decision.**

| The decision | The number |
|---|---|
| How many meals, desks, textbooks | Average daily attendance |
| Which children need a follow-up visit | Proportion below the threshold |
| Whether an intervention worked | Both, because they can move in opposite directions |

That last row is the one to internalise. **An intervention that brings the worst attenders from 40% to 60% moves average daily attendance barely at all and moves the proportion above 90% not at all** — and would be recorded as a failure by either number alone.

```python
struggling = eligible[eligible["mean"] < 0.75]
print(f"students below 75%: {len(struggling)} "
      f"({len(struggling) / len(eligible):.1%})")
print(f"their marks as a share of all marks: "
      f"{struggling['size'].sum() / eligible['size'].sum():.1%}")
```

```
# The students furthest behind are a small share of the rows and the whole
# of the problem.
```

## 3.5 State the threshold, and where it came from

The 90% threshold is a convention, not a standard, and it does most of the work in that 62.1%. Move it to 85% and the figure becomes 78.9%.

```
sensitivity = {f"{t:.0%}": f"{(eligible['mean'] >= t).mean():.1%}"
               for t in (0.75, 0.80, 0.85, 0.90, 0.95)}
print(sensitivity)


# Print the curve, not the point.
```

At 95% it is 39.8% and at 75% it is 90.5%. **Publish the threshold beside the number, every time.** Two reports quoting "the proportion of students regularly attending" at different thresholds produce incomparable figures that both look official — the same failure as the two Food Consumption Score threshold sets, in a different sector.

## 3.6 The denominator decision nobody makes explicitly

`by_student` above was filtered to students with at least twenty marked days. That is a choice and it has to be declared.

**Without it**, a student who enrolled in the last week of term and attended three days out of three appears as a 100% attender, and a student marked twice appears in the tail.

**With it**, you have excluded exactly the late-arriving and early-leaving students, who are the ones an attendance programme most wants to see.

```
print(f"students with any marks:      {len(by_student)}")
print(f"students with 20 or more:     {len(eligible)}")


# Two denominators, both defensible, and the report says which.
```

Here the two are nearly identical, so the choice does not move the answer. **In a register covering a full year it would move it a great deal**, and the habit of declaring it is what makes the figure portable.

## 3.7 Report the pair

```
Attendance, February to April 2024

  Average daily attendance         88.4%   69,974 usable marks
  Students attending 90%+ of days  62.1%   1,200 students with 20+ marks
  Students attending 85%+          78.9%
  Students below 75%                9.5%   114 students, the follow-up list

  SCH09 recorded 873 marks as Y/N rather than true/false; recoded, not
  dropped. 271 days were never marked and are excluded from both figures.
```

## 3.8 What comes next

Both numbers in this lesson divide by days that were marked. The next lesson is about the days that are not in the file at all — fifteen of them, at two schools, which turn a strike into a dropout emergency if you let them.

CHAPTER 4

# A strike that looks like an emergency

Lesson 4 of 8 · 105 min

*SCH07 attends at 88.2% and SCH18 at 85.2%. Build the register as a full calendar and count every missing day as an absence, and they read 66.1% and 63.9% — the worst schools in the district, on fifteen days nobody was meant to be there.*

## 4.1 The rows that are not there

Attendance rows exist only for days a school was open. **A date with no row is a closure, not an absence**, and nothing in the file marks which is which.

```python
import pandas as pd

attendance = pd.read_csv("school-attendance-2024.v1.csv")
roster = pd.read_csv("school-roster-2024.v1.csv")

joined = attendance.merge(roster[["student_id", "school_id"]], on="student_id")
calendar = sorted(attendance["attendance_date"].unique())

days_per_school = joined.groupby("school_id")["attendance_date"].nunique()
print(f"school days in the calendar: {len(calendar)}")
print(days_per_school.sort_values().head())
```

```r
library(dplyr)

attendance |>
  left_join(roster, by = "student_id") |>
  summarise(days = n_distinct(attendance_date), .by = school_id) |>
  arrange(days)
```

**Twenty-two schools have all sixty days. SCH07 and SCH18 have forty-five.**

```python
missing = {
    school: sorted(set(calendar) - set(group["attendance_date"]))
    for school, group in joined.groupby("school_id")
}
for school, dates in missing.items():
    if dates:
        print(f"{school}: {len(dates)} days, {dates[0]} to {dates[-1]}")
```

```r
# Which dates, not just how many. The pattern is the diagnosis.
```

**Both are missing exactly 11 to 29 March**, and both are missing the same fifteen days. That is not two schools with an attendance problem; that is one event.

**Consecutive missing days at multiple schools is a closure until proved otherwise.** Scattered missing days at one school is a recording problem. The shape tells you which, and it takes one line to look.

## 4.2 What happens if you do not look

The natural way to build an attendance table is to construct the full grid of students by school days and fill in the marks. It is also the way to turn a strike into a crisis.

```
def attendance_rate(school, missing_counts_as_absent):
    students = roster.loc[roster["school_id"] == school, "student_id"]
    marks = attendance[attendance["student_id"].isin(students)]
    present = marks["present"].isin(["true", "Y"]).sum()
    if missing_counts_as_absent:
        denominator = len(students) * len(calendar)
    else:
        denominator = marks["present"].isin(["true", "Y", "false", "N"]).sum()
    return present / denominator

for school in ("SCH07", "SCH18", "SCH01"):
    observed = attendance_rate(school, False)
    filled = attendance_rate(school, True)
    print(f"{school}: observed {observed:.1%}, grid-filled {filled:.1%}")
```

```
# Two denominators: marks made, and student-days in the calendar.
```

| School | On marks made | On a full grid |
|---|---|---|
| SCH07 | 88.2% | **66.1%** |
| SCH18 | 85.2% | **63.9%** |
| SCH01 | 91.3% | 91.3% |

**Twenty-two points, invented.** On the grid-filled figures SCH07 and SCH18 are the two worst schools in the district by a wide margin, and a programme reading that table would send a dropout response to two schools whose children attended normally on every day they were asked to.

**The unaffected school is unchanged**, which is what makes the error so hard to catch: the table looks fine, most of it *is* fine, and the two wrong rows are the two you act on.

## 4.3 The rule

**Build the denominator from days the school was open, not from the calendar.**

```
open_days = joined.groupby("school_id")["attendance_date"].nunique()
enrolled = roster.groupby("school_id")["student_id"].nunique()
expected = (open_days * enrolled).rename("student_days_expected")

actual = joined.groupby("school_id").size().rename("marks_made")
coverage = (actual / expected).rename("mark_coverage")
print(pd.concat([expected, actual, coverage.round(3)], axis=1).head())
```

```
# Expected student-days uses each school's own open days.
```

That gives a second, useful number: **mark coverage**, the share of expected student-days that carry a mark at all. A school at 100% attendance and 60% mark coverage is not a school with good attendance.

## 4.4 When a closure is the finding

Fifteen school days is a quarter of the term. **The closure is more consequential than any attendance figure in this file**, and an analysis that correctly excludes those days and then says nothing about them has removed the largest thing that happened.

```
Attendance, February to April 2024

  Average daily attendance, all schools    88.4%   on marks made
  SCH07                                    88.2%
  SCH18                                    85.2%

  SCH07 and SCH18 were closed for the fifteen school days from 11 to 29
  March, a quarter of the term. Their attendance figures are computed on the
  45 days they were open and are comparable with other schools; their
  instructional time is not.

  Counting the closure days as absences would report these two schools at
  66.1% and 63.9% and rank them worst in the district.
```

**Report the closure as lost instructional days, separately from attendance.** The two answer different questions: attendance is about whether children came, and instructional days are about whether school happened.

## 4.5 The general form

This is the third time this platform has met the same defect in a different file.

| Course | The absent thing | What it looked like |
|---|---|---|
| Routine data and DHIS2 | A facility that did not report | A district with falling coverage |
| WASH analysis | A monitoring round nobody drove | A district with improving functionality |
| **Education** | **A day the school was closed** | **Two schools with a dropout emergency** |

**In all three, the absence is not a value and nothing in the file announces it.** The habit that catches all three is the same: before computing any rate, construct what the denominator *should* be from something other than the rows you have, and compare.

## 4.6 What comes next

Attendance describes one term. Whether a child is still in school next year is a different question, and the next unit follows the 2023 cohort through promotion, repetition and dropout to find out.

# Part III

# The cohort

CHAPTER 5

# Three outcomes that have to sum to one

Lesson 5 of 8 · 105 min

*70.4% promoted, 9.8% repeated, 10.2% dropped out, 5.7% finished and 2.9% transferred. Every one of those needs the same denominator, and the transfers are the reason it is not the enrolment count.*

## 5.1 The year, as a set of exits

```python
import pandas as pd

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
clean = enrolment[enrolment["age_years"] <= 20]
cohort = clean[clean["school_year"] == 2023]

outcomes = cohort["end_of_year_status"].value_counts(normalize=True)
print((outcomes * 100).round(1))
print(f"\ncohort: {len(cohort)} students")
```

```r
library(dplyr)

enrolment |>
  filter(age_years <= 20, school_year == 2023) |>
  count(end_of_year_status) |>
  mutate(share = n / sum(n))
```

| Outcome | Students | Share |
|---|---|---|
| Promoted | 910 | 70.4% |
| **Dropped out** | 132 | **10.2%** |
| **Repeated** | 127 | **9.8%** |
| Completed the final grade | 74 | 5.7% |
| Transferred out | 37 | 2.9% |
| Still enrolled at the cut-off | 12 | 0.9% |

**The three headline rates are promotion, repetition and dropout, and they only mean anything if they share a denominator.** That sounds obvious and it is the step most often skipped.

## 5.2 The denominator is not the enrolment count

Two categories complicate it, and they complicate it in opposite directions.

**Transfers are not dropouts.** A child who moved to another school is still in school. Counting them as dropouts overstates dropout by nearly three points and — worse — attributes a system success to a school failure.

**Transfers are also not successes.** They cannot be counted as promoted, because this register does not know whether they enrolled anywhere.

```python
def rates(frame, exclude_transfers):
    base = frame[frame["end_of_year_status"] != "transferred-out"] \
        if exclude_transfers else frame
    return {
        "n": len(base),
        "promotion": (base["end_of_year_status"]
                      .isin(["promoted", "completed-final-grade"]).mean()),
        "repetition": base["end_of_year_status"].eq("repeated").mean(),
        "dropout": base["end_of_year_status"].eq("dropped-out").mean(),
    }

for exclude in (False, True):
    result = rates(cohort, exclude)
    label = "excluding transfers" if exclude else "all students"
    print(f"{label:20} n={result['n']}  "
          f"promotion {result['promotion']:.1%}  "
          f"repetition {result['repetition']:.1%}  "
          f"dropout {result['dropout']:.1%}")
```

```r
cohort |>
  filter(end_of_year_status != "transferred-out") |>
  summarise(n = n(),
            promotion = mean(end_of_year_status %in% c("promoted", "completed-final-grade"
    )),
            repetition = mean(end_of_year_status == "repeated"),
            dropout = mean(end_of_year_status == "dropped-out"))
```

**This is the CMAM cure-rate denominator from earlier in this module, in a different sector.** The convention there was to exclude transfers because the programme cannot claim their outcome; the convention here is the same, for the same reason. **Say which you used, and the three rates will sum to something you can defend.**

## 5.3 Repetition is a decision, not an event

```python
by_district = clean.groupby("admin2")["repeating"].agg(["mean", "size"])
print((by_district * [100, 1]).round(1))
```

```r
enrolment |> filter(age_years <= 20) |>
  summarise(repeating = mean(repeating), n = n(), .by = admin2)
```

| District | Repetition rate | n |
|---|---|---|
| Sud | 12.3% | 583 |
| Artibonite | 12.1% | 618 |
| Nord-Ouest | 12.0% | 625 |
| **Centre** | **7.4%** | 624 |

Centre reports repetition nearly five points below every other district. **That is either the best-performing district in the region or a recording habit, and the register alone cannot tell you which.**

It is a recording habit: Centre flags a share of its genuine repeaters as new entrants. The way to suspect it without being told is that repetition should track over-age, and in Centre it does not — **8.7% repetition against 11.8% elsewhere, while Centre's over-age**

**share is 40.6% against 35.7%.** The district reporting the least repetition has the *most* over-age children, which is the wrong way round for any explanation except the coding.

```
centre = clean[(clean["admin2"] == "Centre") & (clean["school_year"] == 2023)]
elsewhere = clean[(clean["admin2"] != "Centre") & (clean["school_year"] == 2023)]
for name, frame in [("Centre", centre), ("elsewhere", elsewhere)]:
    over = frame["age_years"] > frame["grade"] + 5
    print(f"{name}: repetition {frame['repeating'].mean():.1%}, "
          f"over-age {over.mean():.1%}")


# Repetition should predict over-age. Where it does not, suspect the coding.
```

**A rate that is out of line with the indicator it mechanically causes is a question about the office before it is a finding about the schools.** That is the same reasoning as the protection course's closure-reason catch-all, and it is worth having as a reflex.

## 5.4 Dropout by grade, and the grade-6 spike

```
by_grade = cohort[cohort["grade"].between(1, 6)].groupby("grade").agg(
    n=("student_id", "size"),
    dropout=("end_of_year_status", lambda s: s.eq("dropped-out").mean()),
    repetition=("end_of_year_status", lambda s: s.eq("repeated").mean()),
)
print((by_grade * [1, 100, 100]).round(1))
```

```
cohort |> filter(between(grade, 1, 6)) |>
  summarise(n = n(),
            dropout = mean(end_of_year_status == "dropped-out"),
            repetition = mean(end_of_year_status == "repeated"), .by = grade)
```

| Grade | n | Dropout | Repetition |
|---|---|---|---|
| 1 | 149 | 8.1% | 2.0% |
| 2 | 271 | 8.5% | 7.0% |
| 3 | 332 | 13.0% | 6.6% |
| 4 | 231 | 8.7% | 10.4% |
| 5 | 165 | 8.5% | 19.4% |
| **6** | 110 | **18.2%** | 12.7% |

Two things in that table need reading carefully.

**Repetition rises with grade and dropout does not.** Grade 5 holds back 19.4% of its students while its dropout is 8.5%; grade 6 does the opposite. The two are alternative exits from the same decision point, and reading either alone gets the shape of the problem wrong.

**Grade 6 dropout is 18.2% on 110 students.** That is the largest rate in the table and the smallest denominator, so before treating it as the priority, put an interval on it — the survey course's machinery, on a proportion that would move several points if four children had done something else.

## 5.5 Report the exits, then the rates

```
End of school year 2023, 1,292 students

  Promoted                     70.4%   910
  Dropped out                  10.2%   132
  Repeated                      9.8%   127
  Completed the final grade     5.7%    74
  Transferred out               2.9%    37   excluded from the rates below
  Still enrolled at cut-off     0.9%    12

  Rates on 1,255 students, transfers excluded:
    Promotion (incl. completion)  78.4%
    Repetition                    10.1%
    Dropout                       10.5%

  Centre reports repetition at 7.4% against 12.0-12.3% elsewhere while
  holding the highest over-age share of the four districts. Treat as a
  recording difference pending verification, not as performance.
```

## 5.6 What comes next

These are one year's exits. Turning them into "how many children finish primary school" needs a cohort, and the next lesson builds one — along with the reason the number it produces is almost certainly wrong.

CHAPTER 6

# A survival rate you should not publish

Lesson 6 of 8 · 105 min

*Chain one year's promotion rates and 19.3% of children reach the final grade. Count repeaters as retained and it is 40.3%. Both come from the same file, and neither is a completion rate.*

## 6.1 The calculation everyone reaches for

You have promotion rates by grade for one year. Chain them and you have survival to the final grade — the SDG-style completion measure — without waiting six years.

```python
import pandas as pd

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
clean = enrolment[enrolment["age_years"] <= 20]
cohort = clean[clean["school_year"] == 2023]

survival, cumulative = {}, 1.0
for grade in range(1, 7):
    grade_rows = cohort[cohort["grade"] == grade]
    promoted = grade_rows["end_of_year_status"].isin(
        ["promoted", "completed-final-grade"]).mean()
    survival[grade] = cumulative
    cumulative *= promoted
    print(f"grade {grade}: promotion {promoted:.1%}, "
          f"survival to here {survival[grade]:.1%}")

print(f"\nsurvival to grade 6 entry: {cumulative:.1%}")
```

```r
library(dplyr)

cohort |>
  filter(between(grade, 1, 6)) |>
  summarise(promotion = mean(end_of_year_status %in%
              c("promoted", "completed-final-grade")), .by = grade) |>
  arrange(grade) |>
  mutate(survival = cumprod(lag(promotion, default = 1)))
```

| Grade | Promotion | Cumulative survival |
|---|---|---|
| 1 | 89.3% | 100.0% |
| 2 | 83.0% | 89.3% |
| 3 | 71.7% | 74.1% |
| 4 | 76.2% | 53.1% |
| 5 | 70.9% | 40.5% |
| 6 | 67.3% | 28.7% |
| — | — | **19.3%** |

**19.3% of children entering grade 1 reach the end of primary school.** It is a striking number, it is what the method produces, and it should not leave your screen.

## 6.2 Why it is wrong

This is a **reconstructed cohort**: it assumes that this year's grade-5 promotion rate is what today's grade-1 children will face in four years. Four things break that, and here they break it in the same direction.

**Repeaters are counted as failures.** A child who repeats grade 2 is not out of school — they are in grade 2 again. The chain treats non-promotion as exit, so 10.1% repetition compounds six times into a huge apparent loss.

```
promotion_or_repeat = cohort[cohort["grade"].between(1, 6)].groupby("grade")[
    "end_of_year_status"].apply(
    lambda s: s.isin(["promoted", "completed-final-grade", "repeated"]).mean())

still_enrolled = 1.0
for grade, rate in promotion_or_repeat.items():
    still_enrolled *= rate
print(f"survival counting repeaters as retained: {still_enrolled:.1%}")
```

```
# Retention, not promotion, is what a survival rate needs.
```

**Transfers are counted as failures too.** 2.9% left for another school and the chain reads them as leaving education.

**A single year is assumed to be six.** The grade-6 promotion rate observed in 2023 belongs to children who entered in 2018, under different conditions.

**The grade sizes are not a cohort.** Grade 1 has 149 students and grade 6 has 110, and some of that is genuine attrition while some is a changing population — a larger birth cohort arriving at grade 1 makes the pyramid look like dropout.

## 6.3 What to compute instead

**Retention, not promotion**, if you must use a single year. Count children who are still in school — promoted or repeating — as retained.

```
def chained(statuses):
    rate, product = {}, 1.0
    for grade in range(1, 7):
        rows = cohort[cohort["grade"] == grade]["end_of_year_status"]
        product *= rows.isin(statuses).mean()
        rate[grade] = product
    return product

promotion_only = chained(["promoted", "completed-final-grade"])
retained = chained(["promoted", "completed-final-grade", "repeated"])
print(f"chained promotion: {promotion_only:.1%}")
print(f"chained retention: {retained:.1%}")
```

```
# Same chain, different numerator. The gap is repetition compounding.
```

**19.3% against 40.3%.** More than twenty points of the apparent loss was repetition being read as exit, and the retention figure is the one closer to something meaningful — though still not a completion rate.

**Follow real students where you can.** This register has two years and the same identifiers, so a one-year transition is directly observable rather than assumed.

```
years = clean.pivot_table(index="student_id", columns="school_year",
                          values="grade", aggfunc="first")
both = years.dropna()
print(f"students observed in both years: {len(both)}")
print(f"  advanced a grade: {(both[2024] > both[2023]).mean():.1%}")
print(f"  same grade:       {(both[2024] == both[2023]).mean():.1%}")
```

**1,103 students appear in both years: 88.5% advanced a grade and 11.5% are in the same grade twice.** That is a measured transition, not an assumed one, and it is the number to put in a report.

```
enrolment |>
  select(student_id, school_year, grade) |>
  tidyr::pivot_wider(names_from = school_year, values_from = grade) |>
  filter(!is.na(`2023`), !is.na(`2024`))
```

**A one-year transition observed on real students beats a six-year chain assembled from one year's rates**, even though it answers a smaller question. The smaller question is one you can defend.

## 6.4 Who leaves

The disaggregation that does work, unlike the over-age cuts in lesson 2.

```
year23 = cohort.assign(
    over_age=cohort["age_years"] > cohort["grade"] + 5,
    dropped=cohort["end_of_year_status"].eq("dropped-out"),
)
for column in ("over_age", "disability_reported", "displacement_status", "sex"):
    table = year23.groupby(column, dropna=False)["dropped"].agg(["mean", "size"])
    print((table * [100, 1]).round(1), "\n")
```

```
cohort |>
  mutate(dropped = end_of_year_status == "dropped-out") |>
  summarise(dropout = mean(dropped), n = n(), .by = displacement_status)
```

| Cut | Dropout | n |
|---|---|---|
| **Over-age** | **19.5%** | 478 |
| In-age | 4.8% | 814 |
| Returnee | 19.7% | 71 |
| Disability reported | 17.9% | 112 |
| Internally displaced | 14.7% | 163 |
| Host community | 11.6% | 95 |
| Resident | 8.6% | 940 |
| Boys | 10.4% | 634 |
| Girls | 10.0% | 658 |

**Over-age is the strongest predictor and the largest group.** Disability and displacement roughly double the rate on smaller denominators.

**Sex shows nothing** — 10.4% against 10.0% on 634 and 658 students. That is a result worth stating, because a gender gap in dropout is the finding an education programme most expects to see and this register does not contain one.

## 6.5 The missing data is not missing at random

```python
print(year23["displacement_status"].isna().sum(), "students with no status")
print(year23.loc[year23["displacement_status"].isna(), "dropped"].mean())
```

```r
cohort |> filter(is.na(displacement_status)) |> nrow()
```

Displacement status is blank for 42 students across the register, 23 of them in the 2023 cohort. The blanks are drawn entirely from displaced and returnee households — it is the field an enumerator skips when a family has just arrived.

**Be precise about what that does and does not bias.** The blanks are removed roughly at random *within* those two categories, so:

- the **dropout rate** for internally displaced and returnee students is unbiased — 14.7% and 19.7% are the right numbers for those groups;
- their **counts and population share** are understated by about a tenth. 234 students are recorded as displaced or returnee in 2023 and the true figure is nearer 260.

So "displaced children drop out at 14.7%" is defensible and "displaced children are 18% of enrolment" is not. **A missing-value pattern can bias a count without biasing a rate**, and which one your sentence relies on decides whether the pattern matters.

The observed dropout among the blanks themselves is 8.7% on 23 students, which is too few to read anything into and should not be reported as a category.

## 6.6 Report it as a risk profile

```
Dropout, 2023 cohort, 1,122 students

  Overall                       10.2%

  Over-age for grade            19.5%   478 students
  In-age                         4.8%   814

  Returnee                      19.7%    71 students
  Disability reported           17.9%   112
  Internally displaced          14.7%   163
  Resident                       8.6%   940

  No meaningful difference by sex: 10.4% boys, 10.0% girls.

  42 students have no displacement status, all drawn from displaced and
  returnee households. The rates above are unbiased for those groups; their
  counts are understated by about a tenth.

  Not reported: survival to the final grade. A reconstructed cohort from a
  single year gives 19.3% on promotion and 40.3% on retention, neither of
  which is a completion rate. A true cohort needs six years of the register
  or a household survey. The measured one-year transition is 88.5%.
```

**"Not reported, and why" is the most useful line in that block.** Somebody will ask for a completion rate, and the answer is a real one: not from this, from a household survey, and here is what this can tell you instead.

## 6.7 What comes next

Enrolment, attendance and retention describe whether children are in school. None of them says whether they are learning, and the last unit is the instrument that does — where two rounds turn out not to be comparable.

# Part IV

# Learning

CHAPTER 7

# Eight points of improvement, two of them nobody earned

Lesson 7 of 8 · 105 min

*Literacy rises from 53.1% to 61.8% of items across the two rounds. On the 585 students who sat both, it rises from 56.8% to 63.7%. The difference is who turned up, and the raw scores could not have told you either number.*

## 7.1 The comparison that cannot be made

```python
import pandas as pd

assessment = pd.read_csv("learning-assessment-2024.v1.csv")
literacy = assessment[assessment["domain"] == "literacy"]

print(literacy.groupby("assessment_round").agg(
    students=("student_id", "nunique"),
    items=("items", "first"),
    mean_raw=("raw_score", "mean"),
).round(1))
```

```r
library(dplyr)

assessment |>
  filter(domain == "literacy") |>
  summarise(students = n_distinct(student_id), items = first(items),
            mean_raw = mean(raw_score), .by = assessment_round)
```

| Round | Students | Items | Mean raw score |
|---|---|---|---|
| Baseline | 741 | **40** | 21.2 |
| Endline | 658 | **50** | 30.9 |

**A raw score went from 21.2 to 30.9 and that comparison is meaningless**, because the instruments are not the same length. Nine points of the increase is ten extra questions.

This is the single most common error in education assessment reporting, and it is invisible unless you look at the `items` column — which most extracts do not carry and most analysts do not ask for.

## 7.2 Three things that are comparable, and one that is not

**Not comparable: the raw score.** Different denominators.

**Comparable with care: the proportion of items correct.** It puts both rounds on a 0–1 scale, which is necessary and not sufficient — the endline items may be harder or easier, and a proportion assumes they are equivalent.

```python
literacy = literacy.assign(proportion=literacy["raw_score"] / literacy["items"])
print(literacy.groupby("assessment_round")["proportion"].mean().round(3))
```

```
assessment |> filter(domain == "literacy") |>
  summarise(proportion = mean(raw_score / items), .by = assessment_round)
```

**53.1% at baseline and 61.8% at endline.** Better, and still resting on an assumption nobody has checked.

**Comparable by construction: the proficiency bands.** These were equated when the instruments were designed, which is what equating is for — it is a psychometric procedure that puts two different tests on one scale, and it is the reason the bands exist as a separate column rather than being derived from the score.

```
bands = pd.crosstab(assessment["assessment_round"],
                    assessment["proficiency_band"], normalize="index")
order = ["below-minimum", "minimum", "proficient", "advanced"]
print((bands[order] * 100).round(1))
```

```
assessment |> count(assessment_round, proficiency_band) |>
  mutate(share = n / sum(n), .by = assessment_round)
```

| Band | Baseline | Endline |
|---|---|---|
| Below minimum | 19.3% | **11.1%** |
| Minimum | 30.6% | 23.9% |
| Proficient | 38.2% | 38.9% |
| Advanced | 11.9% | **26.1%** |

**Below-minimum falls from 19.3% to 11.1% and advanced rises from 11.9% to 26.1%.** That is the comparison to report, because it is the only one whose scale was designed to survive a change of instrument.

## 7.3 Who sat the test

```
sat = assessment.groupby("assessment_round")["student_id"].apply(set)
baseline, endline = sat["baseline"], sat["endline"]

print(f"baseline only: {len(baseline - endline)}")
print(f"endline only:  {len(endline - baseline)}")
print(f"both rounds:   {len(baseline & endline)}")
```

```
# Three groups, and only one of them supports a change estimate.
```

**741 sat the baseline, 658 the endline, and 585 sat both.** 156 students who sat the first test did not sit the second.

**That is not a random 156.** Assessment day catches whoever is in the classroom, and the children who are not in the classroom are the ones attending least — who are also, from lesson 3, the ones scoring lowest.

So the endline mean is computed on a group from which the weakest have been disproportionately removed, and it will rise for that reason alone.

## 7.4 Measure the selection instead of assuming it away

The register lets you do the thing that settles it: compute the change on the students who appear in both rounds.

```
panel = literacy.pivot_table(index="student_id", columns="assessment_round",
                             values="proportion")
panel = panel.dropna()

print(f"panel of {len(panel)} students")
print(f"  baseline {panel['baseline'].mean():.1%} → endline {panel['endline'].mean():.1%}
    ")

all_comers = literacy.groupby("assessment_round")["proportion"].mean()
print(f"all comers")
print(f"  baseline {all_comers['baseline']:.1%} → endline {all_comers['endline']:.1%}")
```

```
assessment |> filter(domain == "literacy") |>
  mutate(proportion = raw_score / items) |>
  select(student_id, assessment_round, proportion) |>
  tidyr::pivot_wider(names_from = assessment_round, values_from = proportion) |>
  filter(!is.na(baseline), !is.na(endline)) |>
  summarise(n = n(), baseline = mean(baseline), endline = mean(endline))
```

| | Baseline | Endline | Change |
|---|---|---|---|
| **All comers** | 53.1% | 61.8% | **+8.7 points** |
| **Panel of 585** | 56.8% | 63.7% | **+6.9 points** |

**Two of the 8.7 points are who turned up.** The panel baseline is 3.7 points above the all-comers baseline, which is the direct measurement of the selection: the students who came back were already stronger.

**Report the panel estimate and show the all-comers figure beside it.** The panel is the defensible number; the gap between them is the evidence that the selection was real and the size of it.

## 7.5 What the panel costs

It is not free, and the trade has to be stated.

**The panel is not the population.** 585 of 741 baseline students, and they are the better-attending ones. So +6.9 points is an unbiased estimate of the change *for children who stayed in school and turned up twice*, which is a narrower claim than the one people want.

**It cannot tell you about the 156 who left.** Their learning may have gone anywhere, and the honest report says the estimate does not cover them.

```
left = literacy[literacy["student_id"].isin(baseline - endline)
                & (literacy["assessment_round"] == "baseline")]
print(f"the 156 who did not return scored "
      f"{left['proportion'].mean():.1%} at baseline")
print(f"the 585 who did scored "
      f"{panel['baseline'].mean():.1%}")
```

```
# Quantify the difference rather than asserting it.
```

**The 156 who did not return scored 39.3% at baseline against the panel's 56.8% — seventeen points below.** That is not a subtle selection effect; it is the weakest sixth of the cohort walking out of the endline mean.

**Quantifying the selection is the deliverable**, not eliminating it. An analysis that says "the endline sample was better-attending, and here is by how much" has done its job; one that reports +8.7 points has not.

## 7.6 Report all three numbers

```
Literacy, baseline to endline

  Proficiency bands, all who sat each round
    Below minimum        19.3% → 11.1%
    Advanced             11.9% → 26.1%

  Proportion of items correct
    All comers           53.1% → 61.8%   +8.7 points   n=741, n=658
    Panel of both rounds 56.8% → 63.7%   +6.9 points   n=585

  Instruments differed: 40 items at baseline, 50 at endline. Raw scores are
  not comparable and are not reported. Proficiency bands were equated at
  design; the proportion of items assumes equivalent difficulty.

  156 baseline students did not sit the endline. They scored 39.3% at
  baseline against 56.8% for those who returned, so the all-comers change
  overstates learning by about 1.8 points. The panel estimate covers
  children who sat both rounds and does not cover those 156.
```

## 7.7 What comes next

You now have every education indicator this course can produce — enrolment, attendance, retention and learning — each with a denominator and a caveat. The last lesson is the report they go into, and the four numbers a head teacher can actually act on.

CHAPTER 8

# Four numbers a head teacher can act on

Lesson 8 of 8 · 105 min

*This course produced fourteen indicators on six denominators. A district education report needs all of them; a head teacher needs four, and choosing which four is the analytical work rather than the formatting.*

## 8.1 Count the denominators first

The habit this module has been building, one last time.

| Indicator | Denominator | n |
|---|---|---|
| Gross enrolment ratio | Children aged 6–11 (projected) | 963 |
| Net enrolment ratio | Children aged 6–11 (projected) | 963 |
| Over-age share | Primary enrolees | 1,052 |
| Average daily attendance | **Marks made** | 69,974 |
| Students attending 90%+ | **Students with 20+ marks** | 1,200 |
| Promotion, repetition, dropout | 2023 students excluding transfers | 1,255 |
| One-year transition | Students in both years | 1,103 |
| Learning, all comers | **Students who sat each round** | 741 / 658 |
| Learning, panel | **Students who sat both** | 585 |

**Six different denominators and none of them is "students".** Two are projected populations, two are marks, three are student counts under different rules, and one is a subsample of a subsample.

```
import pandas as pd

denominators = pd.DataFrame([
    ("gross enrolment", "children aged 6-11, projected", 963),
    ("attendance, ADA", "marks made", 69974),
    ("attendance, regular", "students with 20+ marks", 1200),
    ("dropout", "2023 students, transfers excluded", 1255),
    ("learning change", "students who sat both rounds", 585),
], columns=["indicator", "denominator", "n"])
print(denominators)
```

```
# Build it in code, print it above the results, every time.
```

## 8.2 The district report

```
Education, four districts, school year 2023-24

ENROLMENT                                  denominator: 963 projected 6-11
  Gross enrolment ratio           109.2%   1,052 primary enrolees
  Net enrolment ratio              97.4%     938 aged 6-11
  Over-age for grade               36.4%   grade 1 17.0%, grade 6 50.8%
```

```
  Denominator is a 2015 census projected at 2.1% a year; about a sixth of it
  is an assumption. 12 student-years were recorded twice and deduplicated.

ATTENDANCE                                    February to April 2024
  Average daily attendance          88.4%     69,974 marks
  Students attending 90%+ of days   62.1%     1,200 students with 20+ marks
  Students below 75%                 9.5%     114 students

  SCH07 and SCH18 were closed 11-29 March, fifteen school days. Their rates
  are computed on the 45 days they opened. Counting the closure as absence
  would report them at 66.1% and 63.9% and rank them worst.

RETENTION                                     2023 cohort, transfers excluded
  Promotion (incl. completion)      78.4%     n=1,255
  Repetition                        10.1%
  Dropout                           10.5%
  One-year transition observed      88.5%     n=1,103 in both years

  Dropout is 19.5% among over-age students against 4.8% in-age.
  Centre reports repetition at 7.4% against 12.0-12.3% elsewhere while
  holding the highest over-age share; treat as a recording difference.

LEARNING                                      literacy
  Below minimum band        19.3% -> 11.1%
  Advanced band             11.9% -> 26.1%
  Items correct, panel      56.8% -> 63.7%   +6.9 points, n=585
  Items correct, all comers 53.1% -> 61.8%   +8.7 points

  Instruments were 40 and 50 items; raw scores are not comparable and are not
  reported. 156 baseline students did not sit the endline and scored 39.3%
  against 56.8% for those who did.

NOT REPORTED
  Survival to the final grade. A reconstructed cohort gives 19.3% on
  promotion and 40.3% on retention; neither is a completion rate. A true
  cohort needs six years of the register or a household survey.
  Safely managed anything. This is an enrolment register, not a survey of
  children out of school -- every denominator above counts children the
  system already has.
```

## 8.3 The four numbers for a head teacher

A district report is not a school report, and handing a head teacher the table above is a way of ensuring nothing happens.

**Four numbers, each attached to something a school can change.**

| Number | Why this one | The action |
|---|---|---|
| **Students below 75% attendance** | A list, not a rate | Follow up this term |
| **Over-age share in grade 1** | Late entry is fixable at intake | Registration campaign |
| **Repetition rate, this school** | A decision the school makes | Review the criteria |
| **Below-minimum band share** | The children not learning to read | Targeted support |

```
follow_up = (attendance_rates < 0.75).sum()
print(f"students to visit this term: {follow_up}")
```

```
# One number, one list, one action.
```

**Notice what is not on that list.** The gross enrolment ratio is a district planning number and a head teacher cannot move it. Average daily attendance is a system indicator and it hides exactly the children the school should visit. Both belong in the district report and neither belongs on a school's wall.

**Match the indicator to the decision-maker's actual span of control**, which is the same rule the WASH course applied to a mobile team and the protection course to a caseload — and the reason a single dashboard for every audience serves none of them.

## 8.4 The distinction the whole course was for

Enrolled, attending and learning are three different populations, and this course has produced a number for each on the same children.

```
funnel = pd.DataFrame([
    ("enrolled in primary", 1052),
    ("attending 90%+ of days", int(1200 * 0.621)),
    ("proficient or advanced in literacy", int(658 * 0.65)),
], columns=["stage", "students"])
print(funnel)
```

```
# Three stages, three denominators. Do not draw them as one funnel without
# saying so.
```

**Those three cannot be chained into a funnel**, because they are measured on different denominators over different periods — and drawing them as one is the single most common education dashboard error after the enrolment ratio.

What they *do* support is the sentence this course exists to make available: **a system can enrol nearly every child of school age, have most of them present most days, and still leave one in nine unable to read at the minimum level** — and each of those three facts needs its own instrument, its own denominator and its own caveat.

## 8.5 Module 4, in one paragraph

Six sectors, six vocabularies, one set of questions.

**What did I count, over what?** Person-time in epidemiology, visits in WASH, consenting cases in protection, marks in education. No two of them were "the population" and none of them was obvious.

**What else could explain it?** Age structure in a cholera outbreak, a closed road in a water district, a service opening in a protection caseload, a strike in a school register. In every case the absent thing was not a value and nothing in the file announced it.

**What should someone do differently?** The question that decides whether the first two were worth asking, and the one that decides which four numbers reach the person who can act.

## 8.6 What comes next

Module 4 is complete. Module 5 turns from describing programme data to modelling it — regression, forecasting and the dashboards that carry the result — and every course in it runs on the files this module has been building.

# 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/education-programme-analysis

Generated July 28, 2026.