Denominators chained together

A cascade is a sequence in which each step's denominator is the previous step's numerator. 1,850 protection cases become 757 reaching a service, and the useful number is not the 41% — it is which link lost the most.

Pierre Robentz Cassion

Lesson 2 of 8

Public Health and Applied Epidemiology — The denominator is the epidemiology

What this lesson covers

- The shape
- Two ways to express each step
- The first denominator is the hard one
- The immunisation cascade
- Watch the direction of the dropout
- Draw it, but report the table
- What comes next

The shape — Example

```
people living with HIV
  -> diagnosed
    -> on treatment
      -> virally suppressed
```

Two ways to express each step — In Python (cont.)

```
import pandas as pd

referrals = pd.read_csv("protection-referrals-2024.v1.csv")

steps = [
    ("cases", len(referrals)),
    ("consented", (referrals["consent_to_refer"] == True).sum()),
    ("referral made", ((referrals["consent_to_refer"] == True)
                       & (referrals["referral_made"] == True)).sum()),
    ("reached a service", ((referrals["consent_to_refer"] == True)
                           & (referrals["referral_made"] == True)
                           & (referrals["referral_accepted"] == True)).sum()),
]

first = steps[0][1]
previous = None
```

Two ways to express each step — In Python (cont.)

```
for name, count in steps:
    line = f"{name:20} {count:5}  {count / first:6.1%} of all"
    if previous:
        line += f"   {count / previous:6.1%} of previous"
    print(line)
    previous = count
```

Two ways to express each step — In R

```r
library(dplyr)

referrals |>
  summarise(
    cases = n(),
    consented = sum(consent_to_refer),
    referred = sum(consent_to_refer & referral_made),
    reached = sum(consent_to_refer & referral_made & referral_accepted)
  )
```

Two ways to express each step

Step	n	Of all	Of previous
Cases	1,850	100%	—
Consented to referral	1,638	88.5%	88.5%
Referral made	1,142	61.7%	**69.7%**
Reached a service	757	40.9%	66.3%

Two ways to express each step

- Both columns are needed and they say different things
- Rank by conditional loss, act on the largest

Two ways to express each step — In Python

```
losses = pd.DataFrame(steps, columns=["step", "n"])
losses["lost"] = losses["n"].shift(1) - losses["n"]
losses["conditional_loss"] = losses["lost"] / losses["n"].shift(1)
print(losses.dropna().sort_values("conditional_loss", ascending=False))
```

Two ways to express each step — In R

```
tibble::tibble(step = c("consented", "referred", "reached"),
               n = c(1638, 1142, 757), previous = c(1850, 1638, 1142)) |>
  mutate(conditional_loss = (previous - n) / previous) |>
  arrange(desc(conditional_loss))
```

The first denominator is the hard one

- **HIV**: people living with HIV. Nobody counts them; the figure comes from a model, and every "% diagnosed" inherits...
- **TB**: estimated incident cases. Same, and the gap between estimated and notified is the headline finding of most...
- **Protection**: people who experienced an incident. Unknowable, and the reason the referral cascade here starts at...

The first denominator is the hard one — In Python

```
print("cascade base: cases known to the system, not people in need")
```

The first denominator is the hard one — In R

```
# The base of this cascade is cases the system saw. It is not incidence.
```

The first denominator is the hard one

- Say which base you used — A cascade starting from an estimated need and one starting from cases already in the system...

The immunisation cascade — In Python

```
vax = pd.read_csv("vaccination-coverage-2024.v1.csv")
reported = vax[vax["report_submitted"] == True]
doses = reported.groupby("antigen")["doses_administered"].sum()

for first_dose, last_dose in [("penta1", "penta3"), ("mcv1", "mcv2")]:
    dropout = (doses[first_dose] - doses[last_dose]) / doses[first_dose]
    print(f"{first_dose} -> {last_dose}: {doses[first_dose]:,} -> "
          f"{doses[last_dose]:,}, dropout {dropout:.1%}")
```

The immunisation cascade — In R

```
doses <- vax |> filter(report_submitted) |>
  summarise(doses = sum(doses_administered), .by = antigen)
```

The immunisation cascade

- Penta1 to penta3: 13.8% dropout. Measles first to second dose: 22.9%
- It is robust to the denominator — Numerator and denominator come from the same facilities in the same months, so a...
- It measures the mechanism — Coverage rises if more children start; dropout falls only if more children who start also...

Watch the direction of the dropout — In Python

```
if doses["penta3"] > doses["penta1"]:
    print("negative dropout: check the denominator and the period")
```

Watch the direction of the dropout — In R

```r
if (doses$doses[doses$antigen == "penta3"] > doses$doses[doses$antigen == "penta1"])
  warning("negative dropout")
```

Draw it, but report the table — In Python

```
cascade = pd.DataFrame({
    "step": ["Cases", "Consented", "Referral made", "Reached service"],
    "n": [1850, 1638, 1142, 757],
})
cascade["of_all"] = cascade["n"] / cascade["n"].iloc[0]
cascade["of_previous"] = cascade["n"] / cascade["n"].shift(1)
cascade["base"] = "cases known to the system"
```

Draw it, but report the table — In R

```
tibble::tribble(
  ~step,              ~n,
  "Cases",          1850,
  "Consented",      1638,
  "Referral made", 1142,
  "Reached",         757
) |> mutate(of_all = n / first(n), of_previous = n / lag(n))
```

What comes next

- A cascade is a set of counts already aggregated.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/public-health-epidemiology/epi-02-cascades