The gap opens before the referral is made

Cases reporting a disability complete at 26.7% against 46.2%. The gap is not at the service door — consent is identical at 89% and 88.5% — it opens at the gate where a caseworker decides whether to make the referral at all.

Pierre Robentz Cassion

Lesson 4 of 8

What this lesson covers

- Find the gate, not the total
- Which service, and which area
- The gap that matters
- Before publishing it
- Report the gates, not the total
- What comes next

Find the gate, not the total — In Python (cont.)

```
import pandas as pd

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

def cascade(frame):
    consented = frame[frame["consent_to_refer"]]
    made = consented[consented["referral_made"]]
    accepted = made[made["referral_accepted"]]
    reached = accepted[accepted["days_to_first_service"].notna()]
    return pd.Series({
        "n": len(frame), "consented": len(consented), "made": len(made),
        "accepted": len(accepted), "reached": len(reached),
    })

totals = cascade(referrals)
print(totals)
```

Find the gate, not the total — In Python (cont.)

```
print("\nloss at each gate:")
print(f"  consent:  {1 - totals['consented'] / totals['n']:.1%}")
print(f"  referral: {1 - totals['made'] / totals['consented']:.1%}")
print(f"  accepted: {1 - totals['accepted'] / totals['made']:.1%}")
print(f"  service:  {1 - totals['reached'] / totals['accepted']:.1%}")
```

Find the gate, not the total — In R

```r
library(dplyr)

referrals |> summarise(
  n = n(),
  consented = sum(consent_to_refer),
  made = sum(consent_to_refer & referral_made),
  accepted = sum(referral_made & referral_accepted),
  reached = sum(referral_accepted & !is.na(days_to_first_service))
)
```

Find the gate, not the total

Gate	Lost here
Consent	11.5% — a decision, not a loss
Referral made	**30.3%**
Referral accepted	**33.7%**
Reached the service	5.3%

Find the gate, not the total

- The two middle gates lose almost everything — Once a referral is accepted, 94.7% of cases reach the service — the...

Which service, and which area — In Python

```
by_service = referrals.groupby("service_requested").apply(cascade,
                                                          include_groups=False)
by_service["completion"] = by_service["reached"] / by_service["consented"]
print((by_service["completion"] * 100).round(1).sort_values())
```

Which service, and which area — In R

```
referrals |>
  summarise(consented = sum(consent_to_refer),
            reached = sum(referral_accepted & !is.na(days_to_first_service)),
            .by = service_requested) |>
  mutate(completion = reached / consented) |> arrange(completion)
```

Which service, and which area

Service	Completion	Referral made	Accepted
Livelihood support	**20.7%**	49%	47%
Legal	30.0%	61%	50%
Safety and security	33.8%	61%	58%
Psychosocial	53.2%	77%	72%
Health	**57.6%**	81%	76%

Which service, and which area

- Livelihood support completes at a third of the rate health does — and it fails at both middle gates equally — half the. . .

The gap that matters — In Python

```
disability = referrals["disability_reported"].isin([True, "true", "Yes"])
no_disability = referrals["disability_reported"].isin([False, "false", "No"])

for label, mask in [("reported", disability), ("not reported", no_disability)]:
    row = cascade(referrals[mask])
    print(f"{label:14} n={row['n']:>5}  "
          f"consent {row['consented'] / row['n']:.1%}  "
          f"made {row['made'] / row['consented']:.1%}  "
          f"accepted {row['accepted'] / row['made']:.1%}  "
          f"→ {row['reached'] / row['consented']:.1%}")
```

The gap that matters — In R

```
referrals |>
  mutate(disability = disability_reported %in% c("true", "Yes")) |>
  summarise(n = n(), consented = sum(consent_to_refer),
            made = sum(consent_to_refer & referral_made),
            accepted = sum(referral_made & referral_accepted),
            reached = sum(referral_accepted & !is.na(days_to_first_service)),
            .by = disability)
```

The gap that matters

	n	Consent	Referral made	Accepted	Completion
Disability reported	227	89.0%	**54.5%**	**56.4%**	**26.7%**
Not reported	1,623	88.5%	71.9%	67.3%	46.2%

The gap that matters

- Consent is identical — 89.0% against 88.5% — People with disabilities are exactly as willing to be referred
- Locating the gap at a specific gate turns an observation into an action — "Outcomes are worse for people with...

Before publishing it

- The disability column has four values — One area recorded `Yes` and `No` instead of `true` and `false`, so a naive...

Before publishing it — In Python

```
print(referrals["disability_reported"].value_counts(dropna=False))
```

Before publishing it — In R

```
referrals |> count(disability_reported)
```

Before publishing it

- Disability is self-reported and under-reported — 227 of 1,850 is 12.3%, below most population estimates
- Check the cell sizes before disaggregating further — Disability by area by service is exactly the four-way table the...

Report the gates, not the total — Example

Referral pathway, 1,638 consenting cases

Referral made	69.7%
Accepted, of those made	66.3%
Reached service, of accepted	94.7%
End to end, of consenting	43.8%

The pathway fails upstream. Once a referral is accepted 94.7% of cases reach a service; the losses are at referral-making and acceptance.

Cases reporting a disability: 26.7% complete against 46.2%. Consent is identical (89.0% and 88.5%); the gap is entirely at referral-making (54.5% against 71.9%) and acceptance (56.4% against 67.3%).

Disability is self-reported by 12.3% of cases. Non-disclosure places some people in the comparison group, so the gap is a lower bound.

What comes next

- The pathway ends when a case reaches a service.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/protection-and-gbv-data/prot-04-where-the-pathway-leaks