CASSION · COURSE

Protection and GBV Data

Analyse referral, case management and incident data under the safety and confidentiality rules the sector imposes — including the analyses you must decline to publish.

Instructor	Pierre Robentz Cassion
Level	Advanced
Learner hours	16 h
Taught in	Python · R
Updated	July 28, 2026
Sectors	Protection
Standards and methodologies	Core Humanitarian Standard (CHS) · Sphere Standards · UNICEF indicator definitions · OECD DAC evaluation criteria

Read and run the course online at data-analysis.cassion.dev/courses/protection-and-gbv-data

What you will be able to do

- State what a protection dataset must not contain, and justify each omission by what the analysis actually needs
- Assess disclosure risk in a disaggregated table and apply a suppression rule before the table leaves the building
- Build a consent-gated referral pathway and locate the node where the largest share is lost
- Compute caseload per caseworker and relate it to the quality measures that fall as it rises
- Handle a censored case register without biasing time to closure downward
- Say in writing that a case curve measures reporting rather than prevalence, and show the evidence for it

Prerequisites

None. This course assumes no prior programming experience.

Contents

I What not to collect **1**

1 The columns that are not there **2**
1.1 Start with the schema, not the data . . . 2
1.2 The principle, stated as a rule about columns . . . 2
1.3 What the coarsening costs, measured . . . 3
1.4 The one that is genuinely a loss . . . 4
1.5 The 62 blank age bands . . . 4
1.6 What to write in the methodology note . . . 4
1.7 What comes next . . . 5

2 Nineteen cells of one **6**
2.1 The dataset is anonymous and the table is not . . . 6
2.2 Why four and not one . . . 6
2.3 A rule, written as code . . . 7
2.4 What to do instead of publishing the cell . . . 7
2.5 The equity finding survives the rule . . . 8
2.6 The request that has to be refused . . . 8
2.7 What comes next . . . 9

II The pathway **10**

3 The 212 cases that are not a failure **11**
3.1 The pathway has four gates . . . 11
3.2 Consent is not a performance measure . . . 12
3.3 Report all three numbers . . . 12
3.4 Where consent is not the gate . . . 12
3.5 The contradictions to resolve first . . . 13
3.6 What comes next . . . 13

4 The gap opens before the referral is made **14**
4.1 Find the gate, not the total . . . 14
4.2 Which service, and which area . . . 15
4.3 The gap that matters . . . 15
4.4 Before publishing it . . . 16
4.5 Report the gates, not the total . . . 16
4.6 What comes next . . . 17

III The caseload **18**

5 Thirty-three cases each **19**
5.1 Caseload is a stock, not a flow . . . 19
5.2 The number the guidance is written about . . . 19

5.3 What an overloaded caseload does . . . 20
5.4 The establishment error . . . 21
5.5 What caseload is not . . . 21
5.6 Report it as a supervision table . . . 21
5.7 What comes next . . . 22

6 The cases that have not closed are the long ones 23
6.1 The obvious calculation is biased . . . 23
6.2 Fix the impossible durations first . . . 24
6.3 Three honest ways to report it . . . 24
6.4 Censoring is not evenly spread . . . 24
6.5 The closure reason is a judgement . . . 25
6.6 Report it whole . . . 26
6.7 What comes next . . . 26

IV What the numbers mean 27

7 The area whose cases doubled is the one that improved 28
7.1 The curve, and the obvious reading . . . 28
7.2 What a case count is a count of . . . 28
7.3 The test that distinguishes them . . . 29
7.4 Say it in the report, in the sentence next to the number . . . 30
7.5 The consequence people miss . . . 30
7.6 Where prevalence actually comes from . . . 30
7.7 What comes next . . . 31

8 The section that lists what you refused 32
8.1 The report, whole . . . 32
8.2 Why the declined section is in the document . . . 33
8.3 The four sentences that carry the report . . . 33
8.4 The habit this course was for . . . 34
8.5 What comes next . . . 34

About this course

This is the only course on the platform where the correct answer to an analytical question is sometimes **decline to produce it**. That is not a caveat attached to the material; it is the material.

The audience is real. Protection and GBV information managers are asked weekly for a table that would identify a survivor, usually by someone with entirely good intentions who has not thought it through, and the skill being taught is to recognise the request, refuse the table, and offer the thing that answers the underlying question safely.

The course runs on two files. The **referral dataset** has appeared in earlier courses as a joining and disaggregation problem; here it is analysed as what it is, and **what it does not contain is the first lesson**. It gains a **case management register** — caseworker, month opened, month closed, closure reason — because a pathway analysis cannot answer the two questions a supervisor is judged on.

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

38.8% of cases reach a service and 43.8% of consenting cases do, and the difference between those denominators is a person's decision rather than a programme failure. Consent gates the pathway, and counting a non-consenting case as a failure misstates performance and misrepresents the person.

Cases with a reported disability complete at 26.7% against 46.2%. That gap is nineteen points, it is the reason to disaggregate at all, and it is also the finding most likely to be lost in a table too small to publish.

One area's caseworkers carry 33.2 open cases each against another's 14.8, and the overloaded area holds 1.78 case plan reviews per case against 2.45, and loses contact with 38.9% of the cases it closes. Three tables, one cause.

Both Python and R throughout. You should have done module 3 and the earlier module 4 courses — this course assumes you can defend a denominator, read a censored register, and write a limitation as a decision.

Part I

What not to collect

CHAPTER 1

The columns that are not there

Lesson 1 of 8 · 120 min

This dataset has fourteen columns and no name, no contact detail, no free text, no incident date, no location below district and no perpetrator detail. Every one of those absences is a decision, and every one of them can be justified by what the analysis needs.

1.1 Start with the schema, not the data

```python
import pandas as pd

referrals = pd.read_csv("protection-referrals-2024.v1.csv")
print(referrals.columns.tolist())
print(f"{len(referrals):,} cases, {len(referrals.columns)} columns")
```

```r
library(dplyr)

referrals |> glimpse()
```

Fourteen columns for 1,850 protection cases. Now list what a case management system holds and this extract does not:

Not here	Why not
Name, contact details	Never leaves the case management system, under any circumstance
Free-text narrative	Identifying by construction, and no analysis reads it
Incident date	With an area and a category, frequently identifying
Location below admin2	A village plus a category is a person
Exact age	Age band answers every disaggregation question age can
Incident type, perpetrator relationship	Not shared outside the case management agency at all

Each row of that table is a decision someone made, and each one can be defended by naming the analysis it does not prevent. That is the test: not "is this sensitive" but "what would I be unable to compute without it".

1.2 The principle, stated as a rule about columns

The GBV information management principles are usually taught as ethics. They are also, operationally, instructions about a schema:

- **Collect the minimum necessary.** If no indicator needs a field, holding it is pure risk.
- **Share less than you hold.** The analysis extract is not the case file. Each step outward drops fields.
- **Never share incident-level data outside the case management agency.** Aggregate or do not send.

- **Consent governs use.** A person consenting to a referral has not consented to a research dataset.
- **Safety first, always.** Where an analysis and a survivor's safety conflict, the analysis loses.

```
minimum = {
    "who": ["age_band", "sex", "disability_reported"],
    "where": ["admin1", "admin2"],
    "when": ["referral_month"],
    "what": ["case_category", "service_requested"],
    "pathway": ["consent_to_refer", "referral_made", "referral_accepted",
                "days_to_first_service", "case_status"],
}
print({k: len(v) for k, v in minimum.items()})
```

```
# Five groups of fields, each earning its place by an indicator that needs it.
```

Every field in this extract maps to an indicator. `days_to_first_service` exists because the 72-hour clinical standard for GBV health care needs it; `disability_reported` exists because the equity gap is the finding this dataset was built to surface. A field with no indicator behind it should not be in the extract.

1.3 What the coarsening costs, measured

The claim that coarsening is cheap should be tested rather than asserted.

```
print(referrals["age_band"].value_counts(dropna=False))
print(referrals["referral_month"].nunique(), "distinct periods")
print(referrals["admin2"].nunique(), "areas")
```

```
referrals |> count(age_band)
referrals |> summarise(periods = n_distinct(referral_month),
                       areas = n_distinct(admin2))
```

Five age bands, twelve months, six areas. Now ask what an exact age and an exact date would add:

- **Timeliness**: measured in days from referral to service, which is already a duration and does not need a calendar date.
- **Seasonality**: monthly is the finest resolution any protection trend is read at, and a weekly series on 1,850 cases would be noise.
- **Age disaggregation**: the bands are the reporting categories. A child, an adolescent, a working-age adult and an older person are the four distinctions every protection indicator makes.

So the coarsening costs nothing that anyone asked for. That is what makes it defensible, and it is why the argument for it is analytical rather than merely cautious.

1.4 The one that is genuinely a loss

Not every omission is free, and pretending otherwise is how a data protection argument loses credibility.

Dropping incident type means you cannot say which forms of violence the referral pathway serves worst. That is a real analytical loss, and the answer is not that it does not matter. The answer is that the analysis belongs inside the case management agency, run by the people who hold the data, and what leaves is the conclusion rather than the file.

```
print("Question: does the pathway serve some incident types worse than others?")
print("Where it can be answered: inside the case management agency")
print("What leaves the agency: the finding, not the disaggregation")
```

```
# The right answer to "we need incident type" is often "we will run it and
# send you the result", not "no".
```

"You cannot have the file, and here is the answer to your question" is the professional response, and it is available far more often than either a flat refusal or a quiet handover.

1.5 The 62 blank age bands

```
missing = referrals["age_band"].isna()
print(f"{missing.sum()} cases with no age band ({missing.mean():.1%})")
print(referrals.loc[missing, "case_category"].value_counts())
```

```
referrals |> filter(is.na(age_band)) |> count(case_category)
```

Age band is both the most-used disaggregation and the field most often missing in real intake data, because it is asked at a moment when asking is intrusive. **Sixty-two cases is 3.4% and it is not distributed evenly**, so an age- disaggregated table has a smaller denominator than the pathway table beside it and must say so.

The instinct to impute is wrong here for a reason specific to this sector: an imputed age band attached to a real case in a protection dataset is a fabricated attribute of an identifiable person.

1.6 What to write in the methodology note

```
Data handling

  Analysis extract holds 14 fields for 1,850 cases. It contains no name,
  contact detail, free text, incident date, location below admin2, exact age,
  incident type or perpetrator detail.

  Age is recorded in five bands and time in months. Both are sufficient for
  every indicator reported here, and both reduce the risk that a row can be
  matched to a person.

  Incident-level fields remain in the case management system. Analyses
  requiring them are run by the case management agency and reported as
  findings rather than shared as data.
```

```
62 cases (3.4%) have no age band. Age-disaggregated figures are computed on
1,788 cases and are labelled accordingly.
```

Write this before the results, not in an annex. A protection report whose data handling section is at the back has invited every reader to skip the one part that governs what the rest of it is allowed to say.

1.7 What comes next

Removing columns reduces risk and does not eliminate it. The next lesson is what happens when you cross-tabulate the columns that are left, and the point at which a perfectly anonymous dataset produces a table that identifies someone.

CHAPTER 2

Nineteen cells of one

Lesson 2 of 8 · 120 min

Cross-tabulate area, category, age band and sex and this anonymous dataset produces 200 cells, 85 of them holding four cases or fewer and 19 holding exactly one. In a GBV dataset a cell of one is a person, and the table cannot leave the building.

2.1 The dataset is anonymous and the table is not

Every direct identifier is gone. That protects against reading a row and knowing who it is. It does not protect against **counting**.

```python
import pandas as pd

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

cells = referrals.groupby(
    ["admin2", "case_category", "age_band", "sex"], dropna=False
).size()

print(f"{len(cells)} cells")
print(f"  holding 1 case:  {(cells == 1).sum()}")
print(f"  holding 4 or fewer: {(cells <= 4).sum()}")
```

```r
library(dplyr)

referrals |>
  count(admin2, case_category, age_band, sex) |>
  summarise(cells = n(), of_one = sum(n == 1), of_four_or_fewer = sum(n <= 4))
```

200 cells, 85 of them with four cases or fewer, 19 with exactly one.

A cell of one says: in this district, there was one GBV case involving a girl aged 0–11 this year. Anyone who works in that district may know exactly who that is. The dataset never named her; the table did.

2.2 Why four and not one

The instinct is to suppress only the cells of one. That is not enough, for two reasons.

A cell of two identifies both people to each other, and to anyone who knows one of them.

Differencing recovers suppressed cells. If a row total is published and every cell in it but one is published, the missing one is arithmetic.

```python
by_area = referrals.groupby(["admin2", "case_category"]).size().unstack()
print(by_area)
print("\nIf one cell is suppressed and the row total is printed,")
print("the suppressed cell is the total minus the others.")
```

```
referrals |> count(admin2, case_category) |>
  tidyr::pivot_wider(names_from = case_category, values_from = n)
```

So a suppression rule needs a threshold and a secondary suppression: suppress small cells, then suppress enough additional cells that the small ones cannot be recovered by subtraction. A threshold of five is the common floor in this sector; some agencies use ten.

2.3 A rule, written as code

```
THRESHOLD = 5

def suppress(counts, threshold=THRESHOLD):
    """Primary suppression, then a secondary pass so rows cannot be differenced."""
    table = counts.copy()
    small = table < threshold
    table = table.mask(small, other=pd.NA)

    # Secondary: if a row has exactly one suppressed cell, the row total gives
    # it away. Suppress the next smallest cell in that row as well.
    for index, row in table.iterrows():
        if row.isna().sum() == 1:
            remaining = counts.loc[index][~small.loc[index]]
            if len(remaining):
                table.loc[index, remaining.idxmin()] = pd.NA
    return table

print(suppress(by_area))
```

```
suppress <- function(x, threshold = 5) ifelse(x < threshold, NA, x)
# Plus the secondary pass: a row with exactly one NA is not suppressed at all.
```

Write the rule as a function and apply it to every table. A rule applied by eye is applied inconsistently, and the one table it is forgotten on is the one that gets forwarded.

2.4 What to do instead of publishing the cell

Suppression is the last resort. Three better answers come first, and each keeps the information the requester actually wanted.

Aggregate up. Drop one dimension. Area by category has no cell under five; area by category by age by sex has eighty-five.

```
coarse = referrals.groupby(["admin2", "case_category"]).size()
print(f"cells under 5: {(coarse < 5).sum()} of {len(coarse)}")
```

```
referrals |> count(admin2, case_category) |> summarise(small = sum(n < 5))
```

Report the rate, not the count, on a denominator large enough to carry it. "Completion is 31% for cases reporting a disability" is publishable; "3 of 11 cases in Mirebalais" is not.

Answer the question rather than supplying the table. A requester asking for area by age by sex usually wants to know whether services reach children. That is one number on a large denominator, and it is publishable.

2.5 The equity finding survives the rule

The worry is that suppression destroys the analysis. Test it.

```python
def completion(frame):
    consenting = frame[frame["consent_to_refer"]]
    reached = consenting[consenting["referral_accepted"] &
                         consenting["days_to_first_service"].notna()]
    return len(reached), len(consenting), len(reached) / len(consenting)

disability = referrals["disability_reported"].isin([True, "true", "Yes"])
print("reported:    %d/%d = %.1f%%" % completion(referrals[disability]))
print("not reported: %d/%d = %.1f%%" % completion(referrals[~disability]))
```

```r
referrals |>
  filter(consent_to_refer) |>
  summarise(reached = sum(referral_accepted & !is.na(days_to_first_service)),
            n = n(), .by = disability_reported) |>
  mutate(completion = reached / n)
```

26.7% against 46.2%, on denominators of 202 and 1,436. The nineteen-point equity gap — the most important finding in this dataset — is computed on denominators far above any suppression threshold and is entirely publishable.

The analyses that fail the disclosure test are almost never the ones that matter. A four-way disaggregation of 1,850 cases is rarely answering a question anybody asked; it is usually the result of putting every categorical variable into a `groupby` and seeing what comes out.

2.6 The request that has to be refused

Some do have to be refused, and it is worth rehearsing the sentence.

```
Request: case counts by commune, month and incident category.
Response: declined.

  Commune-by-month cells hold zero to three cases. At that resolution a
  count identifies individuals to anyone working in the area, and the
  dataset cannot be re-identified back but a person can be.

  What I can provide instead:
    - the same counts at district by quarter, no cell below five
    - the completion rate and time to service for each commune, on
      annual denominators
    - a written answer to the question the request is for, if you tell me
      what decision it feeds

  If the requirement is genuinely commune-level operational planning, the
  right route is a data sharing agreement with the case management agency,
  not an extract.
```

Offer three alternatives and ask what decision the request feeds. A refusal with no alternative gets escalated and overturned by someone who has not read this lesson; a refusal with three alternatives usually ends with one of them being accepted.

2.7 What comes next

The rules are established. The next unit is the analysis itself — starting with the gate at the front of the pathway that decides who is in the denominator at all, and it is not a data quality decision.

Part II

The pathway

CHAPTER 3

The 212 cases that are not a failure

Lesson 3 of 8 · 120 min

88.5% of these cases consented to a referral. The 212 that did not are outside the performance denominator, because a person declining a referral is exercising a right rather than revealing a service gap — and counting them as failures does two wrong things at once.

3.1 The pathway has four gates

```python
import pandas as pd

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

gates = {
    "cases recorded": len(referrals),
    "consented to referral": referrals["consent_to_refer"].sum(),
    "referral made": (referrals["consent_to_refer"] &
                      referrals["referral_made"]).sum(),
    "referral accepted": (referrals["referral_made"] &
                          referrals["referral_accepted"]).sum(),
    "reached a service": (referrals["referral_accepted"] &
                          referrals["days_to_first_service"].notna()).sum(),
}
for name, count in gates.items():
    print(f"{name:24} {count:>5}")
```

```r
library(dplyr)

referrals |> summarise(
  cases = 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))
)
```

Gate	Cases	Share of the previous gate
Cases recorded	1,850	—
Consented to referral	1,638	**88.5%**
Referral made	1,142	69.7%
Referral accepted	757	66.3%
Reached a service	717	94.7%

This is the cascade from module 4's first course, in a different sector — chained denominators, each gate measured against the one before it. The arithmetic is the same. **The first gate is not.**

3.2 Consent is not a performance measure

The 212 cases that did not consent could be counted three ways, and only one is defensible.

As failures. End-to-end completion becomes 717 / 1,850 = 38.8%. This treats a person's decision as a programme shortfall.

As exclusions. Completion becomes 717 / 1,638 = 43.8% on cases that consented. This is the performance figure.

As a finding in their own right. 11.5% declined, and *why* is a service design question worth asking separately.

```
consenting = referrals["consent_to_refer"]
reached = referrals["referral_accepted"] & referrals["days_to_first_service"].notna()

print(f"on all cases:        {reached.sum() / len(referrals):.1%}")
print(f"on consenting cases: {reached.sum() / consenting.sum():.1%}")
print(f"declined:            {(~consenting).sum()} ({(~consenting).mean():.1%})")
```

```
referrals |> summarise(
  all_cases = mean(referral_accepted & !is.na(days_to_first_service)),
  consenting = sum(referral_accepted & !is.na(days_to_first_service)) / sum(consent_to_
    refer)
)
```

Counting a non-consenting case as a pathway failure does two wrong things at once. It misstates performance by five points, and it records a person's autonomous decision as a defect in the system that offered them a choice.

Both matter and the second matters more. An indicator that treats declining as failure creates pressure on caseworkers to secure consent, which is the opposite of what informed consent means.

3.3 Report all three numbers

```
Referral pathway, 1,850 cases

  Consented to referral            88.5%   1,638
  Declined                         11.5%     212     reported separately
  Reached a service, of consenting 43.8%     717
  Reached a service, of all cases  38.8%             for reference only
```

Publish the consent rate as its own line. A falling consent rate is a signal about trust in the service, and it is invisible if consent is only ever used as a filter.

3.4 Where consent is not the gate

Two situations where this reasoning does not apply, both worth stating so the rule is not over-applied.

Child protection cases involving a young child operate under a best-interests determination rather than the child's consent, and the consent field records the caregiver's decision. The denominator logic is the same and the ethical basis is different.

Life-threatening emergencies proceed to a life-saving referral without waiting for consent to a data transfer. Those cases appear in the pathway and the consent column does not govern them.

```python
by_category = referrals.groupby("case_category")["consent_to_refer"].agg(
    ["mean", "size"]
)
print((by_category * [100, 1]).round(1))
```

```r
referrals |> summarise(consent = mean(consent_to_refer), n = n(),
                       .by = case_category)
```

Check whether the consent rate differs by category before applying one rule to all of them. Where it does, the denominator decision may need to differ too, and the report has to say which cases were treated which way.

3.5 The contradictions to resolve first

```python
impossible = (referrals["days_to_first_service"].notna() &
              ~referrals["referral_accepted"])
no_referral = impossible & ~referrals["referral_made"]
no_consent = impossible & ~referrals["consent_to_refer"]

print(f"time to service but referral not accepted: {impossible.sum()}")
print(f"  of which no referral was made at all:    {no_referral.sum()}")
print(f"  of which there was no consent:           {no_consent.sum()}")
```

```r
referrals |> filter(!is.na(days_to_first_service), !referral_accepted) |>
  count(referral_made, consent_to_refer)
```

Eleven cases record a time to first service on a referral that was never accepted. Six of them show no referral made at all, and one had no consent. These are logical contradictions and they have to be resolved before any completion rate is trusted, because each one is simultaneously a numerator and not a denominator.

The resolution is a judgement and it must be written down. Trusting the service date implies the pathway fields are unreliable; trusting the pathway fields implies the service date is a keying error. **Say which you trusted and how many cases it moved.**

3.6 What comes next

The pathway loses cases at every gate. The next lesson finds the gate where the largest share is lost, and the group of people for whom every gate is worse.

CHAPTER 4

The gap opens before the referral is made

Lesson 4 of 8 · 120 min

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.

4.1 Find the gate, not the total

An end-to-end completion rate of 43.8% tells you the pathway is leaking. It does not tell you where, and the four gates lose very different amounts.

```python
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)
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%}")
```

```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))
)
```

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

The two middle gates lose almost everything. Once a referral is accepted, 94.7% of cases reach the service — the receiving end works. The failure is upstream: a referral that is never made, and a referral that is made and refused.

Those are two different problems with two different owners. **A single completion rate hides which one you have**, and a programme reading 43.8% would reasonably invest in the service that is not the bottleneck.

4.2 Which service, and which area

```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())
```

```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)
```

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%

Livelihood support completes at a third of the rate health does, and it fails at both middle gates equally — half the referrals are never made and half of those made are refused. That is the signature of a service that is known to be oversubscribed: caseworkers stop referring to it because they know the answer.

Areas range from 29.7% to 53.8%, a spread wide enough to act on and narrow enough that no area is fine.

4.3 The gap that matters

```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%}")
```

```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)
```

	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%

Consent is identical — 89.0% against 88.5%. People with disabilities are exactly as willing to be referred.

The gap opens at the two gates the *service system* controls. A referral is made for 54.5% of them against 71.9%, and accepted for 56.4% against 67.3%. By the service door the difference has compounded to nineteen points.

Locating the gap at a specific gate turns an observation into an action. "Outcomes are worse for people with disabilities" is a finding nobody can act on. "Caseworkers make a referral for 55% of them against 72%, and receiving services accept 56% against 67%" names two conversations with two named parties.

4.4 Before publishing it

Three checks, and the first will change your numbers.

The disability column has four values. One area recorded `Yes` and `No` instead of `true` and `false`, so a naive grouping produces four categories, two of them too small to interpret and one of them silently dropped by a boolean filter.

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

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

Disability is self-reported and under-reported. 227 of 1,850 is 12.3%, below most population estimates. The gap is real; the denominator is a group who disclosed, and people who did not disclose are counted in the comparison group. That biases the gap **downward** — so 19 points is a floor.

Check the cell sizes before disaggregating further. Disability by area by service is exactly the four-way table the last lesson refused to publish.

4.5 Report the gates, not the total

```
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.
```

4.6 What comes next

The pathway ends when a case reaches a service. What happens after that is a case that someone has to carry, and the next unit is the register that records who is carrying how many.

Part III

The caseload

CHAPTER 5

Thirty-three cases each

Lesson 5 of 8 · 120 min

Guidance puts an active GBV caseload near 25 cases per worker. One area here averages 33.2 and peaks at 51. It also holds 1.78 case plan reviews per case against 2.45, and loses contact with 38.9% of the cases it closes. Three tables, one cause.

5.1 Caseload is a stock, not a flow

A case count is a flow: how many cases opened this month. A caseload is a stock: how many are open **right now**, and it is the number a caseworker experiences.

Computing it means expanding each case across the months it was open.

```python
import pandas as pd

cases = pd.read_csv("protection-case-management-2024.v1.csv")
CUTOFF = 12

def months_open(row):
    opened = int(row["opened_month"][5:7])
    closed = int(row["closed_month"][5:7]) if pd.notna(row["closed_month"]) else CUTOFF
    return range(opened, max(closed, opened) + 1)

active = [
    {"caseworker_id": row["caseworker_id"], "admin2": row["admin2"], "month": month}
    for _, row in cases.iterrows()
    for month in months_open(row)
]
active = pd.DataFrame(active)

caseload = active.groupby(["caseworker_id", "month"]).size().rename("open_cases")
print(caseload.describe().round(1))
```

```r
library(dplyr)

cases |>
  mutate(opened = as.integer(substr(opened_month, 6, 7)),
         closed = coalesce(as.integer(substr(closed_month, 6, 7)), 12L)) |>
  rowwise() |>
  mutate(month = list(seq(opened, max(closed, opened)))) |>
  tidyr::unnest(month) |>
  count(caseworker_id, month, name = "open_cases")
```

This is the person-time expansion from module 4's first course, applied to a caseworker rather than to a patient. A case open from March to August contributes to six monthly caseloads, and a register read one row at a time never shows it.

5.2 The number the guidance is written about

```
by_area = caseload.reset_index().merge(
    cases[["caseworker_id", "admin2"]].drop_duplicates("caseworker_id"),
    on="caseworker_id",
)
summary = by_area.groupby("admin2")["open_cases"].agg(["mean", "max"]).round(1)
print(summary.sort_values("mean", ascending=False))
```

```
# Mean and peak monthly caseload per area.
```

Area	Mean caseload	Peak
Port-de-Paix	**33.2**	51
Gonaives	28.2	41
Saint-Marc	24.7	43
Hinche	20.7	44
Mirebalais	18.7	32
Saint-Louis-du-Nord	14.8	26

GBV case management guidance puts an active caseload at around 25. Port-de-Paix sits above it all year and peaks at twice it; Saint-Louis-du-Nord sits well below.

Report the mean and the peak. A worker at 51 open cases in one month has, that month, roughly four working hours per case including travel and documentation, and an annual mean of 33 conceals it.

5.3 What an overloaded caseload does

The reason caseload has a threshold is that things fail when it is exceeded, and this register lets you watch two of them fail together.

```
quality = cases.groupby("admin2").agg(
    cases=("case_id", "size"),
    reviews_per_case=("case_plan_reviews", "mean"),
)
closed = cases[cases["closure_reason"].notna()]
quality["lost_contact"] = closed.groupby("admin2")["closure_reason"].apply(
    lambda s: (s == "lost-contact").mean()
)
print(quality.round(2).sort_values("reviews_per_case"))
```

```
cases |>
  summarise(n = n(), reviews = mean(case_plan_reviews),
            lost = mean(closure_reason == "lost-contact", na.rm = TRUE),
            .by = admin2)
```

Area	Caseload	Reviews per case	Lost contact
Saint-Louis-du-Nord	14.8	1.74	26.0%
Port-de-Paix	**33.2**	**1.78**	**38.9%**
Mirebalais	18.7	2.10	30.8%
Gonaives	28.2	2.31	24.7%
Saint-Marc	24.7	2.38	22.1%
Hinche	20.7	2.45	20.2%

Port-de-Paix carries the highest caseload, holds among the fewest case plan reviews, and loses contact with the largest share of the cases it closes. Three different tables produced by one cause, and the caseload table is the one that names the cause.

Note that Saint-Louis-du-Nord also holds few reviews on a low caseload — so reviews per case is not a clean function of caseload, and the honest reading is that Port-de-Paix has a workload problem while Saint-Louis-du-Nord may have a different one. **A pattern that fits three areas and not the fourth is still a pattern, and saying which area does not fit is part of reporting it.**

5.4 The establishment error

```
areas_per_worker = cases.groupby("caseworker_id")["admin2"].nunique()
print(areas_per_worker[areas_per_worker > 1])
```

```
cases |> summarise(areas = n_distinct(admin2), .by = caseworker_id) |>
  filter(areas > 1)
```

One caseworker identifier appears under two areas, because a worker who transferred was re-registered rather than moved.

A caseload computed per worker is right. A caseload computed per worker per area is wrong, because it splits one person's real workload across two rows and makes both look manageable. Decide which the identifier means before you group by it, and if the register cannot tell you, that is a question for the office rather than an assumption for the analyst.

5.5 What caseload is not

It is not a productivity measure. A worker with 40 open cases is not working harder than one with 15; they are more likely to be failing 40 people slowly. Using caseload to rank workers inverts what the indicator is for.

It is not comparable across case types. A child protection case with a case plan involving a school and a guardian is not equivalent to a one-off legal referral, and this register's `case_category` is what lets you weight them.

```
mix = cases.groupby(["admin2", "case_category"]).size().unstack(fill_value=0)
print((mix.div(mix.sum(axis=1), axis=0) * 100).round(1))
```

```
cases |> count(admin2, case_category) |>
  mutate(share = n / sum(n), .by = admin2)
```

Check the case mix before comparing caseloads. An area with more child protection cases is carrying more work per case, and a raw comparison penalises it.

5.6 Report it as a supervision table

```
Caseload, 2024, 1,108 cases across 17 caseworkers

  Area                 Mean   Peak   Reviews/case   Lost contact
```

```
Port-de-Paix          33.2     51          1.78         38.9%
Gonaives              28.2     41          2.31         24.7%
Saint-Marc            24.7     43          2.38         22.1%
Hinche                20.7     44          2.45         20.2%
Mirebalais            18.7     32          2.10         30.8%
Saint-Louis-du-Nord   14.8     26          1.74         26.0%

Guidance: active caseload around 25 cases per worker.
One caseworker identifier appears in two areas after a transfer; caseload
is computed per worker, not per worker per area.

Port-de-Paix is above the guideline all year and shows the pattern that
follows: fewest case plan reviews and most cases closed for lost contact.
```

Put the guideline in the table. A caseload of 33 means nothing to a reader who does not know what 25 is, and the whole point of the row is the comparison.

5.7 What comes next

Cases close, and how long they take is the other number a supervisor is judged on. The next lesson is that measurement, and the 42.6% of cases that have not closed yet and would bias it if they were ignored.

CHAPTER 6

The cases that have not closed are the long ones

Lesson 6 of 8 · 120 min

Median time to closure is four months on the 636 cases that closed. 472 more are still open, 140 of them already past four months, and child protection is 58% censored against general protection's 32% — so the same statistic means different things in different columns.

6.1 The obvious calculation is biased

```python
import pandas as pd

cases = pd.read_csv("protection-case-management-2024.v1.csv")

opened = cases["opened_month"].str[5:].astype(int)
closed = cases["closed_month"].str[5:].astype("Int64")
duration = closed - opened

print(f"closed cases: {duration.notna().sum()}")
print(f"median months to closure: {duration.median()}")
print(f"still open: {duration.isna().sum()} ({duration.isna().mean():.1%})")
```

```r
library(dplyr)

cases |>
  mutate(opened = as.integer(substr(opened_month, 6, 7)),
         closed = as.integer(substr(closed_month, 6, 7)),
         duration = closed - opened) |>
  summarise(closed = sum(!is.na(duration)),
            median = median(duration, na.rm = TRUE),
            open = sum(is.na(duration)))
```

Four months, on 636 of 1,108 cases. The other 472 are still open at the December cut-off, and they are not missing — they are **censored**.

The bias has a direction and you can see it directly.

```python
still_open = 12 - opened[duration.isna()] + 1
print(f"open cases already past 4 months: {(still_open > 4).sum()}")
print(f"longest open: {still_open.max()} months and counting")
```

```r
cases |> filter(is.na(closed_month)) |>
  mutate(so_far = 12 - as.integer(substr(opened_month, 6, 7)) + 1) |>
  summarise(past_four = sum(so_far > 4), longest = max(so_far))
```

140 of the open cases have already been open longer than the median closed case, and one has been open eleven months. Every one of them will close at a duration above four months, or never. Dropping them makes the answer smaller than the truth, systematically.

6.2 Fix the impossible durations first

```python
impossible = duration.notna() & (duration <= 0)
print(f"cases closing on or before the month they opened: {impossible.sum()}")
print(cases.loc[impossible, ["opened_month", "closed_month"]])
```

```r
cases |> filter(!is.na(closed_month),
                closed_month <= opened_month) |> nrow()
```

Seven cases close in a month earlier than they opened. The contradiction is invisible in either column alone and produces negative durations that a `median()` absorbs without complaint.

Decide and record: exclude them, or treat the closing month as a keying error and set it to the opening month. **Seven cases will not move the median; the discipline of finding and declaring them is what will not be there next time if you skip it.**

6.3 Three honest ways to report it

Report the median with the censoring stated. The cheapest correct option.

```
Median time to closure: 4 months (636 closed cases).
472 cases (42.6%) were still open at the cut-off and are excluded; 140 of
them have already been open longer than 4 months, so the true median is
higher.
```

Report a completion-by-month curve. What share of cases opened in month m had closed within k months, computed only on cases with at least k months of follow-up. This is the cohort approach and it uses the censored cases correctly for as long as they were observed.

```python
def closed_within(k):
    eligible = cases[opened <= 12 - k]
    dur = (eligible["closed_month"].str[5:].astype("Int64")
           - eligible["opened_month"].str[5:].astype(int))
    return (dur <= k).sum() / len(eligible), len(eligible)

for k in (3, 6, 9):
    share, n = closed_within(k)
    print(f"closed within {k} months: {share:.1%} (n={n})")
```

```r
# For each k, restrict to cases with k months of possible follow-up.
```

Report time to closure only for a cohort with full follow-up. Cases opened in January have eleven months of observation; cases opened in November have one. Restricting to the early cohort answers a narrower question honestly.

All three are defensible and the first is not enough on its own. A median with no censoring statement beside it is the version that gets quoted.

6.4 Censoring is not evenly spread

```python
by_category = pd.DataFrame({
    "closed": duration.notna().groupby(cases["case_category"]).sum(),
```

```
    "open": duration.isna().groupby(cases["case_category"]).sum(),
})
by_category["censored"] = by_category["open"] / by_category.sum(axis=1)
print(by_category.round(3))
```

```
cases |> summarise(closed = sum(!is.na(closed_month)),
                   open = sum(is.na(closed_month)), .by = case_category) |>
  mutate(censored = open / (open + closed))
```

Category	Closed	Open	Censored	Median of the closed
Child protection	131	179	**58%**	5 months
GBV	255	173	40%	4 months
General protection	250	120	32%	4 months

Child protection is 58% censored and general protection 32%. So the two medians in that table are computed on very different fractions of their cohorts, and the gap between five months and four months is an understatement of the real gap.

Uneven censoring makes a comparison of medians misleading in a specific direction: the group with more censoring is the one whose median is most understated, which is the group already looking worse.

6.5 The closure reason is a judgement

```
print(cases["closure_reason"].value_counts(normalize=True).round(3))
```

```
cases |> filter(!is.na(closure_reason)) |> count(closure_reason) |>
  mutate(share = n / sum(n))
```

Reason	Share of closures
Case plan objectives met	29.1%
Lost contact	27.2%
Survivor withdrew	13.1%
Closed administratively	11.3%
Relocated	9.7%
Transferred to another agency	9.6%

Only 29.1% of closures are a case plan completed. That is the headline, and it is the number a supervision conversation starts from.

But look at how the reasons distribute by area before believing any of it.

```
by_area = pd.crosstab(cases["admin2"], cases["closure_reason"], normalize="index")
print((by_area[["closed-administratively", "lost-contact"]] * 100).round(1))
```

```
cases |> filter(!is.na(closure_reason)) |>
  count(admin2, closure_reason) |> mutate(share = n / sum(n), .by = admin2)
```

Hinche files 36.2% of its closures as `closed-administratively` against 3.8% to 9.8% everywhere else, and its lost-contact rate reads 20.2% against up to 38.9%. The catch-all is absorbing the reason a supervisor needs.

A closure reason is a caseworker's judgement, not an observation. So a distribution that differs sharply between offices is a question about the offices before it is a question about the cases — and the fix is a coding conversation, not a statistical adjustment.

6.6 Report it whole

```
Case closure, 1,108 cases

  Closed by the cut-off              636    57.4%
  Still open                         472    42.6%    censored, not missing
  Median months to closure             4              of closed cases only
    open cases already past 4 months  140               so the true median is higher

  Censoring by category: child protection 58%, GBV 40%, general 32%.
  Medians are not comparable across those columns.

  Closure reasons (of 636 closures)
    Case plan objectives met       29.1%
    Lost contact                   27.2%
    Survivor withdrew              13.1%
    Closed administratively        11.3%    36.2% in Hinche, 3.8-9.8% elsewhere
    Relocated                       9.7%
    Transferred                     9.6%

  7 cases record a closing month before their opening month and are excluded.
```

6.7 What comes next

Everything in this unit is about cases already in the system. The last unit asks what the number of cases means at all — and why an area whose case count doubled is usually the area where something went right.

Part IV

What the numbers mean

CHAPTER 7

The area whose cases doubled is the one that improved

Lesson 7 of 8 · 120 min

Saint-Louis-du-Nord's monthly intake goes from 5.8 cases to 10.2 after July. Nothing about violence in that area changed. A service opened, and a case curve measures whether people can reach a desk — never how many of them needed to.

7.1 The curve, and the obvious reading

```python
import pandas as pd

cases = pd.read_csv("protection-case-management-2024.v1.csv")
monthly = cases.groupby(["admin2", "opened_month"]).size().unstack(fill_value=0)
print(monthly)
```

```r
library(dplyr)

cases |> count(admin2, opened_month) |>
  tidyr::pivot_wider(names_from = opened_month, values_from = n, values_fill = 0)
```

One area's line goes up and stays up.

```python
area = cases[cases["admin2"] == "Saint-Louis-du-Nord"]
month = area["opened_month"].str[5:].astype(int)
before = (month <= 6).sum() / 6
after = (month >= 7).sum() / 6
print(f"Jan-Jun: {before:.1f} cases a month")
print(f"Jul-Dec: {after:.1f} cases a month  ({after / before - 1:+.0%})")
```

```r
cases |>
  filter(admin2 == "Saint-Louis-du-Nord") |>
  mutate(half = if_else(as.integer(substr(opened_month, 6, 7)) <= 6, "H1", "H2")) |>
  count(half) |> mutate(per_month = n / 6)
```

5.8 cases a month before July, 10.2 after — a 74% rise, while no other area moves by more than eleven points either way.

The obvious reading is that protection incidents in Saint-Louis-du-Nord rose 74% in the second half of the year. **That reading is wrong, and it is the single most common misuse of protection data.**

7.2 What a case count is a count of

A case exists when a person reached a service, was willing to disclose, and a caseworker opened a file. So the count is a product of four things, only the last of which is what people read it as:

The count depends on	Which changes when
A service existing within reach	One opens, one closes, a road is cut
People knowing it exists	An outreach campaign runs
People being willing to disclose	Trust rises or falls; a bad experience circulates
Someone opening a file	Staffing, workload, referral practice
Incidence in the population	—

Every one of the first four moved in Saint-Louis-du-Nord in July, because a service opened. The fifth is not measured by this file and cannot be.

```
print("Case count = incidence x reporting rate x service availability x recording")
print("A change in the product does not tell you which factor moved.")
```

```
# One equation, five unknowns, one observation.
```

7.3 The test that distinguishes them

You cannot prove the curve is a reporting effect from the curve alone. You can gather evidence, and three checks are usually available.

Did anything change in the service? The strongest evidence and the least statistical. A new service, a new outreach worker, a changed intake form or a partner closing all produce a step change with a date attached.

Did the composition change? Saint-Louis-du-Nord's general protection share goes from 31.4% to 41.0% while GBV falls from 40.0% to 32.8% — the new arrivals skew toward the least stigmatised category, which is what improved access looks like rather than what a surge in violence looks like.

```
def mix(frame):
    return frame["case_category"].value_counts(normalize=True).round(3)

print("before July:", mix(area[month <= 6]).to_dict())
print("after July: ", mix(area[month >= 7]).to_dict())
```

```
cases |> filter(admin2 == "Saint-Louis-du-Nord") |>
  mutate(half = if_else(as.integer(substr(opened_month, 6, 7)) <= 6, "H1", "H2")) |>
  count(half, case_category) |> mutate(share = n / sum(n), .by = half)
```

Did other areas move? A rise confined to one area with an administrative explanation is a reporting effect. A rise across every area at the same time is more likely to be real, or a change in the reporting system.

```
halves = cases.assign(half=(cases["opened_month"].str[5:].astype(int) > 6))
shift = halves.groupby(["admin2", "half"]).size().unstack()
shift["change"] = shift[True] / shift[False] - 1
print((shift["change"] * 100).round(1).sort_values())
```

```
cases |> mutate(half = as.integer(substr(opened_month, 6, 7)) > 6) |>
  count(admin2, half) |> tidyr::pivot_wider(names_from = half, values_from = n)
```

One area up 74.3%; the other five range from −9.4% to +11.3%. A single area moving seven times as far as the widest of the others has an administrative cause, and the

job is to find it rather than to publish an incidence claim.

7.4 Say it in the report, in the sentence next to the number

```
Cases opened, Saint-Louis-du-Nord

  Jan–Jun    35 cases    5.8 per month
  Jul–Dec    61 cases   10.2 per month   +74%

  A case management service opened in the area in July. This figure measures
  the number of people who reached a service and disclosed; it is not a
  measure of the number of incidents. The rise is consistent with improved
  access and cannot be read as a rise in violence.

  The other five areas moved between −9% and +11% over the same period.
```

The disclaimer belongs beside the number, not in a footnote. A protection case curve without that sentence will be quoted as an incidence trend within a week, usually by someone quoting your report accurately.

7.5 The consequence people miss

If a case count measures access, then **a programme that succeeds should see its case count rise**, and a falling case count is ambiguous at best.

```
outcomes = pd.DataFrame({
    "observation": ["cases rise", "cases fall"],
    "good reading": ["outreach and access improved",
                     "incidents fell"],
    "bad reading": ["violence increased",
                    "service became unreachable, or trust was lost"],
})
print(outcomes)
```

```
# The same number supports opposite conclusions. The context decides.
```

So a protection programme cannot use case volume as a performance indicator in either direction, and the indicators that survive are the ones this course has been building: pathway completion, time to service, caseload, closure reasons. Each of those has a denominator that is not the population.

7.6 Where prevalence actually comes from

Named plainly, because the question will be asked.

Population-based prevalence surveys, designed for the purpose, with specialised interviewer training and ethical protocols that a service dataset does not have. They are expensive, infrequent and the only instrument that answers the question.

Never from service data, at any level of sophistication. No adjustment for reporting rate rescues a denominator that is "people who reached a service".

```
print("Question: how many women in this district experienced violence this year?")
print("Answerable from this dataset: no")
print("Instrument that answers it: a population-based prevalence survey")
```

```
# Saying "we cannot answer that with this" is the answer.
```

7.7 What comes next

The last lesson assembles everything into a protection report — including the section that states which analyses were requested and declined, and why that section belongs in the document rather than in an email nobody keeps.

CHAPTER 8

The section that lists what you refused

Lesson 8 of 8 · 120 min

A protection report needs a block naming the analyses that were requested and declined, and what was offered instead. It belongs in the document rather than in an email, because the request will be made again by someone who has not read the email.

8.1 The report, whole

```
Protection and GBV data, 2024
Three admin1 areas, six admin2 areas

DATA HANDLING                                          read this first
  Analysis extract: 14 fields, 1,850 cases. No name, contact detail, free
  text, incident date, location below admin2, exact age, incident type or
  perpetrator detail. Age in five bands, time in months.
  Suppression rule: no cell below 5, with secondary suppression so rows
  cannot be differenced.
  Incident-level analyses remain with the case management agency and are
  reported as findings, not shared as data.

REFERRAL PATHWAY                           1,638 consenting cases
  Consented to referral            88.5%   1,638 of 1,850
  Declined                         11.5%     212   a decision, not a failure
  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 accepted, 94.7% reach a service.
  Livelihood support completes at 20.7% against health at 57.6%.

EQUITY
  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, so the gap is a lower bound.

CASE MANAGEMENT                            1,108 cases, 17 caseworkers
  Mean caseload, highest area       33.2   peak 51; guidance is about 25
  Mean caseload, lowest area        14.8
  Case plan reviews per case, highest-caseload area   1.78 against 2.45
  Closed for lost contact, highest-caseload area     38.9% against 20.2%

  Closed by the cut-off              636   57.4%
  Still open                         472   42.6%   censored, not missing
  Median months to closure             4            of closed cases only;
                                                    140 open cases already
                                                    exceed it
  Case plan objectives met         29.1%   of closures

WHAT THESE NUMBERS ARE NOT
  Case counts measure access and reporting, not incidence. Saint-Louis-du-Nord
```

```
  opened a service in July and its monthly intake rose 74%; no conclusion
  about violence in that area follows.
  Prevalence is not estimable from service data at any level of
  sophistication. It requires a population-based survey.

REQUESTED AND DECLINED
  Case counts by commune, month and incident category -- declined; cells of
  0 to 3 identify individuals. Offered: district by quarter, and commune-level
  completion rates on annual denominators.
  Full case list with age and area for a partner's targeting exercise --
  declined; the extract is not a beneficiary list and consent does not cover
  it. Offered: aggregate caseload by area to inform their staffing.

DATA QUALITY
  62 cases (3.4%) have no age band; age-disaggregated figures use 1,788.
  11 cases record a service time on an unaccepted referral; excluded, and
  the completion rate is computed without them.
  7 cases close before they open; excluded from duration statistics.
  One caseworker identifier spans two areas after a transfer; caseload is
  computed per worker.
  One area records disability as Yes/No rather than true/false; normalised.
```

8.2 Why the declined section is in the document

Because the request will be made again. By a new information manager, by a donor's consultant, by a colleague who was not copied on the email. A refusal that lives in a mailbox has to be re-argued every time, usually by someone junior, under time pressure, to someone senior.

Because it shows the refusal was reasoned rather than reflexive. A report that lists two declined requests alongside what was offered instead reads as professional judgement. A report with no such section, from an analyst who says no in meetings, reads as obstruction.

Because it is a design brief. Two years of declined requests is a specification for what the information system should be able to produce safely, and nobody collects that unless it is written down somewhere durable.

```
declined = [
    {"request": "counts by commune, month, incident category",
     "reason": "cells of 0-3 identify individuals",
     "offered": "district by quarter; commune completion rates"},
    {"request": "full case list with age and area",
     "reason": "not a beneficiary list; consent does not cover it",
     "offered": "aggregate caseload by area"},
]
```

```
# Keep the log as data. It is the only way it survives a staff change.
```

8.3 The four sentences that carry the report

Everything above is evidence. A reader takes away four things, and they should be written before the tables rather than extracted from them.

The pathway fails before the service door. 94.7% of accepted referrals reach a service; the losses are at referral-making and acceptance, which are two conversations with two named parties.

People with disabilities are failed by the system, not by their own choices. Identical consent, nineteen points of completion gap, entirely at the two gates the service system controls.

One area is carrying a third more cases than the guideline and it shows in two other tables. Caseload is the cause, and reviews per case and lost contact are the symptoms.

No number here measures how much violence occurred. Every denominator in this report is people who reached a service.

8.4 The habit this course was for

Three questions, in the platform's usual order, with one addition specific to this sector.

1. **What did I count, over what?** Consenting cases, accepted referrals, open cases, closures — four denominators and none of them is the population.
2. **What else could explain it?** A service opening, a coding habit, a caseworker's judgement about what "administratively closed" means.
3. **What should someone do differently because of this?** Two conversations about referral-making, one about establishment in Port-de-Paix, one about closure coding in Hinche.
4. **Who could be harmed if this were published as it stands?** The question the other three do not ask, and the one that governs whether the answer to them ever leaves the building.

The fourth question is the whole of this course, and it is the only one on this platform where the right answer is sometimes to produce nothing.

8.5 What comes next

Module 4 has one course left. Education attendance and learning outcomes bring a new sector and, for the first time in this module, a dataset the platform does not yet have — which is a content problem before it is an analytical one.

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/protection-and-gbv-data

Generated July 28, 2026.