CASSION · COURSE

WASH Analysis

Water access, quality and water point functionality against the JMP service ladders and the Sphere indicators, including the chlorination residual nobody logs consistently.

Instructor	Pierre Robentz Cassion
Level	Intermediate
Learner hours	16 h
Taught in	Python · R
Updated	July 28, 2026
Sectors	WASH
Standards and methodologies	Sphere Standards · Sustainable Development Goals (SDG) · Core Humanitarian Standard (CHS)

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

What you will be able to do

- Place a household on all three JMP service ladders, and say why an improved source is not the same as basic service
- Compute quantity, collection time and queue against the Sphere minimum standards, with the unit errors those fields carry
- Report water quality from a tested subsample without letting it borrow the access denominator
- Handle a left-censored measurement as data rather than dropping it or coercing it to zero
- Produce three defensible functionality rates from one monitoring register and state what each is about
- Separate a water point that fails every dry season from one that has been abandoned, using the visit sequence

Prerequisites

None. This course assumes no prior programming experience.

Contents

About this course

WASH is where this platform's standing instruction — state the numerator, the denominator and the decision — meets a sector that has published definitions for almost everything and a habit of reporting them loosely.

The course runs on two files. The **WASH household survey** has appeared in three earlier courses as a cleaning problem and a denominator problem; here it is finally analysed as what it is, a service-level survey. The **water point monitoring register** is new, and it is the only file on the platform with repeat visits to the same asset — which is what a functionality rate needs and what a household survey structurally cannot provide.

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

Four fifths of households are on an improved source and barely half have basic service. 82.3% against 56.3%, and the twenty-six points between them are collection time: an improved source more than thirty minutes away is *limited* service, and an analysis that classifies on source type alone will report the first number and call it access.

Functionality is three numbers, not one. 74.4% of monitoring visits find a point running; 33.9% of points are running at every visit they receive; 83.3% of the people served have a working point. All three are correct and they are statements about different things.

The best-looking district is the least trustworthy number. Nord-Ouest reads 76.4%, the highest of the three, and made only 82.7% of its due visits because the roads close in the rains. If every missed visit had found a broken point it would be 63.2% — a band 13.2 points wide, against 2.6 for the district that visited nearly everything.

Both Python and R throughout. You should have done module 3 — this course assumes you can defend a denominator, read a routine figure through its reporting rate, and write an indicator reference sheet before you write the analysis.

Part I

The ladder is not the source

CHAPTER 1

Four fifths improved, half with service

Lesson 1 of 8 · 120 min

82.3% of these households are on an improved source and 56.3% have at least basic drinking water service. The twenty-six points between the two are collection time, and they are the whole reason the ladder has rungs.

1.1 The number most WASH reports give

Count the households on an improved source, divide by the households surveyed, and publish it as access.

```python
import pandas as pd

households = pd.read_csv("wash-household-survey-2024.v1.csv")

IMPROVED = {                        # the JMP list, delivered water included
    "piped-into-dwelling", "piped-into-yard", "public-tap",
    "borehole", "protected-well", "protected-spring", "tanker-truck",
}
improved = households["water_source"].isin(IMPROVED)
print(f"improved source: {improved.mean():.1%} of {len(households)} households")
```

```r
library(dplyr)

improved <- c("piped-into-dwelling", "piped-into-yard", "public-tap",
              "borehole", "protected-well", "protected-spring",
              "tanker-truck")

households |> summarise(improved = mean(water_source %in% improved), n = n())
```

82.3%. It is a correct calculation of a thing nobody asked about.

1.2 The ladder the SDG indicator actually uses

The JMP drinking water ladder has four rungs, and the source type only decides which two you are choosing between.

Rung	Definition
Safely managed	An improved source on premises, available when needed, and free from contamination
Basic	An improved source, round trip 30 minutes or less including queueing
Limited	An improved source, round trip **more** than 30 minutes
Unimproved	An unprotected well or spring
Surface water	River, dam, lake, pond, canal, irrigation channel

SDG indicator 6.1.1 is the **safely managed** rung. The humanitarian reporting line, and the one most programmes are measured on, is **at least basic** — safely managed plus basic.

```python
def ladder(row):
    if row["water_source"] == "surface-water":
        return "surface water"
    if row["water_source"] not in IMPROVED:
        return "unimproved"
    minutes = row["round_trip_minutes"]
    return "basic" if minutes <= 30 else "limited"

households["service"] = households.apply(ladder, axis=1)
print(households["service"].value_counts(normalize=True).round(3))
```

```python
households |>
  mutate(service = case_when(
    water_source == "surface-water" ~ "surface water",
    !water_source %in% improved ~ "unimproved",
    round_trip_minutes <= 30 ~ "basic",
    TRUE ~ "limited"
  )) |>
  count(service) |>
  mutate(share = n / sum(n))
```

Rung	Households	Share
Basic (including on premises)	1,352	**56.3%**
Limited	626	26.1%
Unimproved	326	13.6%
Surface water	99	4.1%

At least basic drinking water service: 56.3%. The source-type figure was 82.3%, and every one of the twenty-six points between them is a household on an improved source it takes more than half an hour to reach and return from.

1.3 Why the thirty minutes is in the definition

It is not administrative tidiness. A round trip over thirty minutes changes behaviour in ways that are measurable elsewhere in this same file:

- households collect **less** water, so quantity falls below the Sphere minimum more often;
- collection is done by women and girls, so the time comes out of school attendance and paid work;
- households supplement from a nearer, worse source, so the recorded source is not the only source.

A definition that looks bureaucratic usually encodes a finding. Before arguing that a threshold is arbitrary, look for the thing it was set to separate.

1.4 The rung this survey cannot compute

Safely managed needs three conditions and this file supports one of them.

```python
on_premises = (households["round_trip_minutes"] == 0)
tested = households["ecoli_cfu_100ml"].notna()
```

```
print(f"on premises: {on_premises.mean():.1%}")
print(f"water quality tested: {tested.mean():.1%}")
print(f"both: {(on_premises & tested).mean():.1%}")
```

```
households |> summarise(
  on_premises = mean(round_trip_minutes == 0),
  tested = mean(!is.na(ecoli_cfu_100ml))
)
```

27.7% of households have water on premises. Availability when needed is not asked. Freedom from contamination is tested on a third of the sample. So **safely managed is not computable here, and the honest report says so** rather than reporting on-premises as though it were the top rung.

That is the commonest way an SDG figure gets overstated: one of three conditions is measured, and the indicator is published under the name of all three.

1.5 Report the distribution, not the top rung

```
Drinking water service level, 2,403 households

  At least basic         56.3%   1,352
    of which on premises 27.7%     665
  Limited                26.1%     626
  Unimproved             13.6%     326
  Surface water           4.1%      99

  JMP ladder. Safely managed not computable: availability was not asked and
  water quality was tested on 33.4% of households.
```

Every rung, with counts. A report that gives only "56.3% have basic service" hides that a quarter of the population is one threshold away and would move with a nearer water point — which is exactly the group a programme can act on.

1.6 What comes next

Water has one ladder and two more are waiting. The next lesson does sanitation and hygiene, where the equivalent of the thirty-minute rule is a single boolean column that quietly halves a coverage figure.

CHAPTER 2

The boolean that halves a coverage figure

Lesson 2 of 8 · 120 min

62.0% of households have an improved sanitation facility and 43.2% have basic service. The difference is one column — whether the latrine is shared — and the hygiene ladder has an equivalent that costs it half again.

2.1 Sanitation has the same shape as water

An improved facility is necessary and not sufficient, and the field that makes the difference is a boolean nobody looks at twice.

Rung	Definition
Safely managed	Improved, not shared, and excreta safely disposed of or treated
Basic	Improved, not shared with other households
Limited	Improved but **shared**
Unimproved	Pit latrine without a slab, hanging latrine, bucket
Open defecation	No facility

```python
IMPROVED_SANITATION = {
    "flush-to-sewer", "flush-to-septic", "vip-latrine", "pit-latrine-with-slab",
}

improved = households["sanitation_facility"].isin(IMPROVED_SANITATION)
shared = households["shared_sanitation"]

print(f"improved facility:  {improved.mean():.1%}")
print(f"basic (not shared): {(improved & ~shared).mean():.1%}")
print(f"limited (shared):   {(improved & shared).mean():.1%}")
```

```r
improved_sanitation <- c("flush-to-sewer", "flush-to-septic",
                         "vip-latrine", "pit-latrine-with-slab")

households |>
  mutate(service = case_when(
    sanitation_facility == "open-defecation" ~ "open defecation",
    !sanitation_facility %in% improved_sanitation ~ "unimproved",
    shared_sanitation ~ "limited",
    TRUE ~ "basic"
  )) |>
  count(service) |>
  mutate(share = n / sum(n))
```

Rung	Households	Share
Basic	1,038	**43.2%**
Limited (shared)	452	18.8%
Unimproved	594	24.7%
Open defecation	319	**13.3%**

62.0% have an improved facility and 43.2% have basic service. Eighteen points sit on a latrine that meets the construction standard and is shared with another household.

2.2 Why sharing costs a rung

Because the reason for the standard is use, not construction. A shared latrine is used less at night, less by women and girls, and less by children — which is where the health benefit is. The classification follows the behaviour, not the concrete.

This is the same argument as the thirty minutes. In both ladders the second condition exists because a facility that is technically adequate and practically avoided produces no health outcome, and an indicator that counted it would report progress that nobody experienced.

2.3 Open defecation is the number to lead with

```python
od = households["sanitation_facility"].eq("open-defecation")
by_district = households.groupby("district")["sanitation_facility"].apply(
    lambda s: s.eq("open-defecation").mean()
)
print(f"{od.sum()} households, {od.mean():.1%}")
print((by_district * 100).round(1))
```

```r
households |>
  summarise(open_defecation = mean(sanitation_facility == "open-defecation"),
            n = n(), .by = district)
```

13.3%, and it is the one WASH figure that a non-specialist reads correctly without being told the definition. Lead with it, disaggregate it, and note that the district field in this file is written six ways — the cleaning course dealt with that, and grouping without normalising splits the worst district in four.

2.4 The hygiene ladder, and the observation that carries it

Basic hygiene service requires **a handwashing facility with water and soap available**. Three rungs, and the survey answers all three, because the enumerator observed rather than asked.

```python
has_facility = households["handwashing_facility"].ne("no-facility")
soap = households["soap_observed"]

print(f"any facility:            {has_facility.mean():.1%}")
print(f"basic (facility + soap): {(has_facility & soap).mean():.1%}")
print(f"limited (no soap):       {(has_facility & ~soap).mean():.1%}")
print(f"no facility:             {(~has_facility).mean():.1%}")
```

```r
households |>
  mutate(hygiene = case_when(
    handwashing_facility == "no-facility" ~ "no facility",
    soap_observed ~ "basic",
    TRUE ~ "limited"
  )) |>
  count(hygiene) |>
```

```
  mutate(share = n / sum(n))
```

Rung	Households	Share
Basic — facility and soap	819	**34.1%**
Limited — facility, no soap	811	33.7%
No facility	773	32.2%

67.8% have somewhere to wash their hands and 34.1% have soap there. The hygiene ladder loses half its top rung to a single observed item, which is a larger drop than either of the other two ladders.

Observation is why this is trustworthy. Reported handwashing runs far above observed handwashing everywhere it has been measured, because the question has an obviously correct answer. The column here is what the enumerator saw, and the column name should say so — `soap_observed`, not `soap`.

2.5 The three ladders cannot be averaged

```
summary = pd.DataFrame({
    "ladder": ["Drinking water", "Sanitation", "Hygiene"],
    "at_least_basic": [0.547, 0.432, 0.341],
    "denominator": ["all households"] * 3,
})
print(summary)
```

```
tibble::tribble(
  ~ladder,          ~at_least_basic,
  "Drinking water",           0.547,
  "Sanitation",               0.432,
  "Hygiene",                  0.341
)
```

Three numbers on the same denominator, which is unusual and worth noticing — they can legitimately be shown side by side. What they cannot be is combined. There is no "WASH coverage" indicator, and a mean of the three would describe no household: a household with basic water, shared sanitation and no soap is not "58% covered", it is a household with one of three services.

Where a single composite is demanded, publish the three and the count of households that have all three, which is a real quantity and usually a sobering one.

2.6 What comes next

The ladders classify. They do not say how much water arrived, how long the queue was, or whether the water was safe to drink. The next two lessons are the measurements Sphere adds on top — and the first of them has two unit errors hiding in it.

Part II

What Sphere asks that a ladder does not

CHAPTER 3

Fifteen litres, thirty minutes, five hundred metres

Lesson 3 of 8 · 120 min

13.4% of these households fall below the Sphere minimum for quantity. Two of the columns that produce that number carry unit errors, and one of them moves a household up a ladder rung rather than off the end of a chart.

3.1 What Sphere adds to the ladder

The JMP ladders classify a service. The Sphere minimum standards put numbers on it, and the three that a water programme is judged on are:

Standard	Threshold	In this file
Quantity	At least 15 litres per person per day	`litres_per_person_day`
Queueing time	No more than 30 minutes	Inside `round_trip_minutes`
Distance	No more than 500 m to the nearest point	Not recorded; time is the proxy

The thresholds are minimums for survival with dignity, not targets. **A programme at 15.2 litres has met the standard and is not doing well**, and a report that prints only the share above the threshold hides where the distribution actually sits.

```python
import pandas as pd

households = pd.read_csv("wash-household-survey-2024.v1.csv")
litres = households["litres_per_person_day"]

print(f"median {litres.median():.1f} L/p/d")
print(f"below Sphere minimum: {(litres < 15).mean():.1%}")
print(litres.describe().round(1))
```

```r
library(dplyr)

households |> summarise(
  median = median(litres_per_person_day),
  below_15 = mean(litres_per_person_day < 15),
  n = n()
)
```

13.4% below 15 litres, with a median of 23.7. Both numbers belong in the report: the first is the standard, the second says the median household has about half again what it needs and the problem is distribution rather than total supply.

3.2 Two unit errors, and only one of them is visible

```python
print(litres.nlargest(8).round(1).tolist())
```

```
households |> slice_max(litres_per_person_day, n = 8) |>
  select(household_id, household_size, litres_per_person_day)
```

Eleven households are above 80 litres per person per day, up to 277.6. That is implausible in this setting and it is not a keying error — it is the household's **total** consumption entered in a per-person column. Divide by household size and every one of them lands in a normal range.

```
suspect = households[litres > 80]
recovered = suspect["litres_per_person_day"] / suspect["household_size"]
print(recovered.round(1).describe())
```

```
households |>
  filter(litres_per_person_day > 80) |>
  mutate(recovered = litres_per_person_day / household_size) |>
  summarise(min = min(recovered), max = max(recovered))
```

The second error is the dangerous one, because it makes a value smaller rather than larger. Eleven households have collection time entered in **hours**, so a 90-minute round trip is recorded as 2. There is no outlier to catch: 2 minutes is a perfectly ordinary value, and 85 households legitimately report under 8 minutes.

```
short = households[households["round_trip_minutes"].between(1, 8)]
print(f"{len(short)} households report a round trip of 1 to 8 minutes")
print(short["water_source"].value_counts())
```

```
households |> filter(between(round_trip_minutes, 1, 8)) |> count(water_source)
```

An error that moves a value into the normal range cannot be found by looking for outliers. It has to be found by a rule that ties two fields together — a round trip under ten minutes to a source that is not on premises and not a public tap in the same community deserves a query — or it has to be prevented at entry with a unit label on the form.

Its consequence here is not a wrong mean. It is a **wrong rung**: each of those households is classified as basic service when it is limited.

3.3 Queueing is inside the round trip, and that is a problem

Sphere sets queueing separately from collection time because they have different causes and different fixes. A long walk needs a new water point; a long queue needs more taps at the point that exists, or a longer opening time.

This survey records only the round trip, which contains both. So the honest report says what it can and cannot separate:

```
print("round_trip_minutes includes walking, queueing and return.")
print(f"over 30 minutes: {(households['round_trip_minutes'] > 30).mean():.1%}")
print("Queue time alone: not collected. Recommend adding it to the next round.")
```

```
# One line in the limitations section beats a queue statistic that is a walk.
```

39.3% exceed thirty minutes, and none of that number can be attributed to queueing or to distance without the split. **Say which of the two you cannot see**, because the

corrective action differs and a programme that guesses will build the wrong thing.

3.4 Distance is not in the file at all

The 500-metre standard needs a coordinate or a measured distance, and this survey has neither. Time is a proxy for it and a poor one: thirty minutes is a kilometre on a road and three hundred metres up a ravine.

Where a standard names a unit you do not have, report the proxy under the proxy's own name. Reporting round-trip time as though it were the distance standard is how an assessment concludes that a community is within 500 metres of water when it is nowhere near it.

3.5 Report the distribution against the threshold

```
Water quantity, 2,403 households

  Median                        23.7 L/person/day
  Below Sphere minimum (15 L)  13.4%    323 households
  Below 7.5 L                   2.2%     54 households

  Collection time
  Over 30 minutes              39.3%    944 households
  Queue time not separated from walking time in this round.
  Distance not recorded; time is the proxy and is not equivalent.

  11 records held household total litres in the per-person column and were
  corrected; 11 held collection time in hours and were corrected.
```

The threshold, the distribution, and the corrections stated with counts. A cleaning log is not a confession — it is what lets a reader see that 13.4% is after two known errors were fixed rather than before.

3.6 What comes next

Quantity and time say whether water arrived and how hard it was to get. They say nothing about whether it was safe to drink, and the next lesson is the one where the denominator changes under you.

CHAPTER 4

Tested on a third, censored on a fifth

Lesson 4 of 8 · 120 min

E. coli was tested on 33.4% of households, chlorine on 40.0%, and both on 13.3%. In the water point register, 271 chlorine readings are the string "<0.1" — and dropping them raises apparent compliance by sixteen points.

4.1 The denominator changes, and the report usually does not

Water quality is expensive to measure. A survey that interviews 2,403 households tests a fraction of them, and every quality indicator therefore runs on a different denominator from every access indicator.

```python
import pandas as pd

households = pd.read_csv("wash-household-survey-2024.v1.csv")

ecoli = households["ecoli_cfu_100ml"].notna()
chlorine = households["free_residual_chlorine_mgl"].notna()

print(f"E. coli tested:   {ecoli.sum():>5} = {ecoli.mean():.1%}")
print(f"chlorine tested:  {chlorine.sum():>5} = {chlorine.mean():.1%}")
print(f"both:             {(ecoli & chlorine).sum():>5} = {(ecoli & chlorine).mean():.1%}"
    )
```

```r
library(dplyr)

households |> summarise(
  ecoli = mean(!is.na(ecoli_cfu_100ml)),
  chlorine = mean(!is.na(free_residual_chlorine_mgl)),
  both = mean(!is.na(ecoli_cfu_100ml) & !is.na(free_residual_chlorine_mgl))
)
```

802 households for E. coli, 961 for chlorine, and 320 for both. The last number is the one that matters: any question relating the two runs on 13.3% of the survey, and an interval on it will be wide.

Print the analysable n for every quality figure, in the table rather than in a footnote. A quality percentage next to an access percentage on the same row reads as the same denominator, and here it never is.

4.2 The JMP risk classes

```python
tested = households[ecoli]
classes = pd.cut(
    tested["ecoli_cfu_100ml"],
    [-1, 0, 10, 100, 10**9],
    labels=["safe <1", "low 1-10", "moderate 11-100", "high >100"],
)
print(classes.value_counts(normalize=True).round(3))
```

```
households |>
  filter(!is.na(ecoli_cfu_100ml)) |>
  mutate(risk = cut(ecoli_cfu_100ml, c(-1, 0, 10, 100, Inf),
                    labels = c("safe", "low", "moderate", "high"))) |>
  count(risk) |> mutate(share = n / sum(n))
```

Risk class	Households	Share of tested
Safe (<1 CFU/100 mL)	430	53.6%
Low (1–10)	173	21.6%
Moderate (11–100)	139	17.3%
High (>100)	60	7.5%

Report the classes, not a mean. E. coli counts are heavily skewed and a mean of 14 CFU/100 mL describes nothing — the classes are what the guideline is written in and what a decision is taken on.

4.3 Improved is not safe, and this is where you can prove it

```
IMPROVED = {"piped-into-dwelling", "piped-into-yard", "public-tap",
            "borehole", "protected-well", "protected-spring"}

tested = tested.assign(improved=tested["water_source"].isin(IMPROVED))
print(tested.groupby("improved")["ecoli_cfu_100ml"].agg(
    n="size", safe=lambda s: (s < 1).mean(), high=lambda s: (s > 100).mean()
).round(3))
```

```
households |>
  filter(!is.na(ecoli_cfu_100ml)) |>
  mutate(improved = water_source %in% improved) |>
  summarise(n = n(), safe = mean(ecoli_cfu_100ml < 1),
            high = mean(ecoli_cfu_100ml > 100), .by = improved)
```

Source	Tested	Safe	High risk
Improved	625	**61.1%**	3.8%
Unimproved or surface	177	27.1%	20.3%

Source type predicts quality strongly, and **38.9% of households on an improved source still have detectable E. coli**. Lesson 1 said the top rung needs freedom from contamination as well as an improved source; this is the number that shows why it is a separate condition rather than a formality.

4.4 The censored value, and why it is not missing

The water point register measures chlorine in the field, and the kit has a detection limit.

```
points = pd.read_csv("water-point-monitoring-2024.v1.csv")
readings = points["free_residual_chlorine_mgl"].dropna()

censored = readings.astype(str).str.startswith("<")
print(f"tested {len(readings)}, censored {censored.sum()}")
print(readings[censored].unique())
```

```
points |>
  filter(!is.na(free_residual_chlorine_mgl)) |>
  count(censored = startsWith(free_residual_chlorine_mgl, "<"))
```

811 visits carry a reading and **271 of them are the string `<0.1`**. That is a measurement: the true value lies between zero and the detection limit. It is not a missing value, and the two things it is usually turned into are both wrong.

```
numeric = pd.to_numeric(readings, errors="coerce")   # <-- the silent one
print(f"after coercion: {numeric.isna().sum()} became NaN")

in_range = readings[~censored].astype(float).between(0.2, 0.5)
print(f"in target range, over all tested:   {in_range.sum() / len(readings):.1%}")
print(f"in target range, dropping censored: {in_range.mean():.1%}")
```

```
points |>
  filter(!is.na(free_residual_chlorine_mgl),
         !startsWith(free_residual_chlorine_mgl, "<")) |>
  summarise(in_range = mean(between(as.numeric(free_residual_chlorine_mgl), 0.2, 0.5)))
```

30.9% against 46.5%. Dropping the censored readings raises apparent compliance with the chlorination target by nearly sixteen points, and it does it in exactly one direction — every censored value is a failure, so removing them removes only failures.

`pd.to_numeric(..., errors="coerce")` does this silently and looks like a type fix.

4.5 What to do with a censored value

Three options, in order of preference for this indicator.

Classify rather than average. The target is a range, so a censored reading answers the question perfectly: `<0.1` is below 0.2 and therefore non-compliant. Compliance is computable on all 811 readings with no substitution at all.

Substitute the limit, or half of it, if you genuinely need a mean. State which you used — `LOD/2` is the common convention — and report how many values it applied to.

Report the share below the limit as its own number. 33.4% of these readings are below detection, which is a finding about chlorination practice rather than a nuisance in the data.

```
compliant = pd.to_numeric(readings.where(~censored), errors="coerce").between(0.2, 0.5)
result = pd.DataFrame({
    "readings": [len(readings)],
    "below detection": [censored.sum()],
    "in 0.2-0.5 range": [compliant.sum()],
    "compliance": [f"{compliant.sum() / len(readings):.1%}"],
})
print(result)
```

```
# The denominator is every reading, including the ones below the limit.
```

Never let a detection limit shrink a denominator. It is the single most common way a water quality report becomes optimistic, and it survives review because the arithmetic on the remaining values is correct.

4.6 The cross-field inconsistency worth naming

```
treats = households["water_treated_at_home"]
print(f"treat at home: {treats.sum()}")
print(f"  of which no chlorine reading: {(treats & ~chlorine).sum()}")
```

```
households |> filter(water_treated_at_home) |>
  count(no_reading = is.na(free_residual_chlorine_mgl))
```

533 households report treating their water and have no chlorine measurement. The cleaning course established what this is — missingness that correlates with the thing being measured — and the consequence here is specific: **dropping those rows removes households that treat their water more often than average**, so the tested subsample is not representative of the survey it came from.

4.7 Report it as its own block

```
Water quality

  E. coli, point of collection          802 households tested (33.4% of survey)
    Safe <1 CFU/100 mL                  53.6%
    Low 1-10                            21.6%
    Moderate 11-100                     17.3%
    High >100                            7.5%
    Improved sources: 61.1% safe (n=625)
    Unimproved and surface: 27.1% safe (n=177)

  Free residual chlorine, water points  811 visits tested (30.8% of visits)
    Below detection limit (<0.1 mg/L)   33.4%   counted as non-compliant
    Within 0.2-0.5 mg/L target          30.9%

  Quality was tested on a subsample and does not share the denominator of the
  access indicators above. 533 households reporting home treatment have no
  chlorine reading, so the tested subsample over-represents untreated water.
```

Its own block, its own denominators, its own limitation. **A quality figure placed in a table of access figures will be read against their denominator**, and nothing in the layout will stop it.

4.8 What comes next

Everything so far is a cross-section: what was true on the day someone called. The next unit is the register that visits the same water points twelve times, and the first thing it does is turn one functionality rate into three.

Part III

Service is a year, not a day

CHAPTER 5

74.4%, 33.9%, 83.3%

Lesson 5 of 8 · 120 min

Three functionality rates from one register, all correct. They differ because they count visits, points and people, and only one of them answers the question a water programme is actually asked.

5.1 What a repeat-visit register makes possible

Everything in the first unit is a cross-section. A household survey can tell you what service existed on the day the enumerator called, and it structurally cannot tell you how much of the year the water was there.

The monitoring register can, because it visits the same points again.

```python
import pandas as pd

points = pd.read_csv("water-point-monitoring-2024.v1.csv", parse_dates=["visit_date"])

print(f"visits: {len(points):,}")
print(f"water points: {points['water_point_id'].nunique()}")
print(points["functional_status"].value_counts())
```

```r
library(dplyr)

points |> summarise(visits = n(), water_points = n_distinct(water_point_id))
points |> count(functional_status)
```

2,629 visits to 242 points across twelve monthly rounds. That structure supports three different functionality rates, and the difference between them is not a methodological quibble — it is three different questions.

5.2 Rate one: the share of visits that found a working point

```python
WORKING = {"functional", "partially-functional"}
working = points["functional_status"].isin(WORKING)

print(f"visit-level functionality: {working.mean():.1%} of {len(points):,} visits")
```

```r
points |> summarise(functionality = mean(functional_status %in%
                    c("functional", "partially-functional")), n = n())
```

74.4%. This is what almost every water point report publishes, and it answers *"if I visit a point at random, will it be working?"*

It is the easiest to compute and the easiest to bias, because the denominator is visits made rather than visits due. Lesson 7 is entirely about that.

Partially functional counts as working. A point running at reduced yield is providing water, and classifying it as failure would put a queue problem in the same category as a dry

borehole. Say which side of the line you put it, because reasonable analysts differ and the two answers are three points apart.

5.3 Rate two: the share of points that worked every time

```python
by_point = points.assign(ok=working).groupby("water_point_id")["ok"]
always = by_point.all()

print(f"point-level functionality: {always.mean():.1%} of {len(always)} points")
print(f"points that failed at least once: {(~always).sum()}")
```

```r
points |>
  summarise(always = all(functional_status %in%
            c("functional", "partially-functional")), .by = water_point_id) |>
  summarise(share = mean(always), n = n())
```

33.9% — 82 of 242 points. Two thirds of the network failed at least once during the year.

This answers a different question: *"how many of these assets are reliable?"* And it is the number an asset manager needs, because a point that works three visits in four is not three quarters of a water supply — it is a water supply with a gap in it that a household has to solve some other way.

The two rates are not in tension. 74.4% of visits and 33.9% of points are both true, of the same file, at the same time. A point can be working most of the time and still fail during the year, and a network can be mostly up and mostly unreliable.

5.4 Rate three: the share of people with a working point

```python
served = points.dropna(subset=["users_estimated"])
population_weighted = (
    served.loc[served["functional_status"].isin(WORKING), "users_estimated"].sum()
    / served["users_estimated"].sum()
)
print(f"population-weighted functionality: {population_weighted:.1%}")
```

```r
points |>
  filter(!is.na(users_estimated)) |>
  summarise(weighted = sum(users_estimated[functional_status %in%
            c("functional", "partially-functional")]) / sum(users_estimated))
```

83.3%, nearly nine points above the visit-level figure. The reason is in the source types.

```python
print(points.groupby("source_type").agg(
    points=("water_point_id", "nunique"),
    median_users=("users_estimated", "median"),
    functionality=("functional_status", lambda s: s.isin(WORKING).mean()),
).round(3))
```

```r
points |>
  summarise(n = n_distinct(water_point_id),
            median_users = median(users_estimated, na.rm = TRUE),
            functionality = mean(functional_status %in%
```

```
            c("functional", "partially-functional")), .by = source_type)
```

Source type	Functionality	Typical users
Piped scheme tap	**97.2%**	about 1,290
Handpump borehole	75.5%	about 280
Protected spring	66.6%	about 200
Protected well	**58.1%**	about 185

The points that serve the most people break the least. So weighting by population moves the figure up, and the gap between 74.4% and 83.3% is a real statement about who bears the failures: the people on protected wells, who are the fewest per point and the worst served.

5.5 Which one to report

All three, and this is one of the cases where three numbers is genuinely the right answer.

```
Water point functionality, 2024

  Visit-level        74.4%   1,957 of 2,629 monitoring visits
  Point-level        33.9%      82 of 242 points working at every visit
  Population-weighted 83.3%   of an estimated 96,700 users

  Partially functional counted as working (134 visits).
  Two points were re-registered after a handover and are counted twice;
  removing them moves the visit-level figure to 74.7%.
```

If you must give one, give the population-weighted rate and say so, because the standard is about people rather than assets. But publish the point-level rate next to it, because it is the one that says the network is not reliable, and the population-weighted rate hides that behind a handful of well-run schemes.

5.6 The two points counted twice

```
duplicates = points[points["water_point_id"].str.startswith("WP09")]
print(duplicates.groupby("water_point_id")["community"].agg(["nunique", "first"]))
```

```
points |> filter(startsWith(water_point_id, "WP09")) |> count(water_point_id)
```

Two points were handed from one programme to another and re-registered under new identifiers, so the register holds them twice and every denominator is two points too large. The effect is small — 74.4% becomes 74.7% — and **the size of the effect is not the reason to fix it**. An asset register that double-counts is wrong about what exists, and the next thing anyone does with it is plan a maintenance budget.

5.7 What comes next

Two thirds of these points failed at least once. That number treats a borehole that ran dry in February exactly like one that has been abandoned since March, and the next lesson separates them — using the sequence of visits rather than the status field, because the status field cannot tell them apart.

CHAPTER 6

Dry is not broken

Lesson 6 of 8 · 120 min

Protected wells run at 74.7% in the wet months and 36.4% in the dry ones. Handpump boreholes barely move. The status field cannot tell those two failures apart, and the sequence of visits can.

6.1 One status, several different failures

`functional_status` says a point is not working. It does not say whether the pump is broken, the water table has dropped, the committee has stopped collecting fees, or the community has moved. Those need four different responses and one of them is not a repair.

The sequence of visits can separate them, because they have different shapes over time.

Shape over the year	What it is	What to do
Down in the same months every year, up in between	**Seasonal** — the source runs dry	Deepen, or pro
Down once, then up after a gap	**Breakdown** — repaired	Look at the le
Down and stays down to the end	**Abandoned or awaiting parts**	Find out whic
Up and down repeatedly	**Chronic** — under-maintained or overloaded	Management p

6.2 The seasonal signal, and where it is not

```python
import pandas as pd

points = pd.read_csv("water-point-monitoring-2024.v1.csv", parse_dates=["visit_date"])
WORKING = {"functional", "partially-functional"}
DRY_MONTHS = {1, 2, 3, 11, 12}

points["dry_season"] = points["visit_date"].dt.month.isin(DRY_MONTHS)
points["ok"] = points["functional_status"].isin(WORKING)

print(points.groupby("dry_season")["ok"].agg(["mean", "size"]).round(3))
```

```r
library(dplyr)

points |>
  mutate(dry = lubridate::month(visit_date) %in% c(1, 2, 3, 11, 12),
         ok = functional_status %in% c("functional", "partially-functional")) |>
  summarise(functionality = mean(ok), n = n(), .by = dry)
```

67.8% in the dry months against 79.4% in the wet ones. An eleven-point swing, which a report comparing a March assessment to a July one would read as a collapse or a recovery depending on which way round it did the subtraction.

Now split it by source type, which is where the finding actually is.

```
seasonal = (
```

```
    points.groupby(["source_type", "dry_season"])["ok"].mean().unstack()
)
seasonal.columns = ["wet", "dry"]
seasonal["swing"] = seasonal["wet"] - seasonal["dry"]
print((seasonal * 100).round(1).sort_values("swing", ascending=False))
```

```
points |>
  summarise(functionality = mean(ok), .by = c(source_type, dry)) |>
  tidyr::pivot_wider(names_from = dry, values_from = functionality)
```

Source type	Wet	Dry	Swing
Protected well	74.7%	**36.4%**	38.3
Protected spring	84.8%	**41.0%**	43.8
Handpump borehole	75.0%	76.1%	−1.1
Piped scheme tap	95.9%	98.8%	−2.9

Two source types carry the entire seasonal effect and two do not move at all. Shallow wells and springs draw on a water table that drops; a borehole reaches below it and a piped scheme has a storage tank. The eleven-point average was a mixture of a forty-point effect and no effect.

That is the finding. **Boreholes in this network do not have a seasonal problem and wells do**, so a dry-season programme that rehabilitates boreholes is rehabilitating the wrong asset.

6.3 Classifying each point from its own sequence

```
def classify(group):
    group = group.sort_values("visit_date")
    failures = group.loc[~group["ok"]]
    if failures.empty:
        return "never failed"
    if not group["ok"].tail(3).any() and len(group) >= 3:
        return "down at year end"
    if failures["dry_season"].all():
        return "seasonal"
    return "intermittent"

shape = points.groupby("water_point_id").apply(classify, include_groups=False)
print(shape.value_counts())
```

```
points |>
  arrange(visit_date) |>
  summarise(
    shape = case_when(
      all(ok) ~ "never failed",
      !any(tail(ok, 3)) & n() >= 3 ~ "down at year end",
      all(dry[!ok]) ~ "seasonal",
      TRUE ~ "intermittent"
    ), .by = water_point_id) |>
  count(shape)
```

Shape	Points
Never failed	82
Intermittent	80
Seasonal	46
Down at year end	34

The order of the tests matters and it is a judgement, not a detail. A point that fails only in dry months and is still down in December satisfies both rules. Testing permanence first calls it abandoned; testing seasonality first calls it seasonal. This code tests permanence first, because a point that has not run for three visits is an operational problem now whatever caused it — and because the December reading is the one a January plan is written against.

Write down which order you used. Two analysts with the same file and different orderings will produce different counts and both will be right.

6.4 The status value that looks like the answer and is not

The register has an `abandoned` status, and it is tempting to use it directly.

```
ever_abandoned = points.loc[points["functional_status"].eq("abandoned"),
                            "water_point_id"].nunique()
print(f"points ever recorded as abandoned: {ever_abandoned}")
print(f"points down at the end of the year: 34")
```

```
points |> filter(functional_status == "abandoned") |>
  summarise(points = n_distinct(water_point_id))
```

Seventeen points are recorded as abandoned at some visit; thirty-four are still down at the end of the year. **The status is an enumerator's judgement made at one visit, and the sequence is evidence.** Half the points that never come back were never labelled abandoned, because the enumerator who saw them in July had no way to know they would still be down in December.

Derive the state from the history where you can, and use the recorded status as a cross-check. A field an enumerator fills in from a single observation cannot carry information that only exists across observations.

6.5 Downtime is a management statistic

```
down = points.loc[~points["ok"] & points["days_since_breakdown"].notna()]
print(down.groupby("management")["days_since_breakdown"].agg(
    median="median", visits="size"
).sort_values("median"))
```

```
points |>
  filter(!ok, !is.na(days_since_breakdown)) |>
  summarise(median_days = median(days_since_breakdown), n = n(), .by = management)
```

Management	Median days down at the visit
Private operator	29
Utility	32
NGO-managed	48
Community committee	**68**

A broken point under a community committee has been down more than twice as long as one under a private operator. **This is the one number in the register that points at an action rather than at an asset** — the difference is a spare parts supply chain and a maintenance fund, not the pump.

Beware the thirty-seven rows where a point is recorded as functional and carries a `days_since_breakdown` anyway: it is last month's answer carried forward. Filter on the status, not on the presence of the field.

6.6 Report the shapes, not just the rate

```
Water point failure, 2024, 242 points

  Never failed                82    33.9%
  Intermittent                80    33.1%
  Seasonal (dry months only)  46    19.0%
  Down at year end            34    14.0%

  Seasonality is concentrated in protected wells (74.7% wet, 36.4% dry) and
  protected springs (84.8% wet, 41.0% dry). Handpump boreholes and piped
  schemes show no seasonal effect.

  Points classified as down at year end if not working at any of the last
  three visits; seasonal if every recorded failure fell in a dry month.
  Permanence tested before seasonality.
```

Four counts and the rule that produced them. **A single functionality percentage would send a rehabilitation budget to the wrong forty-six points**, because they do not need rehabilitating — they need a dry-season alternative, and their pumps are fine.

6.7 What comes next

Every number in this lesson is computed over visits that were made. One district missed a third of its rounds in the rainy season, and the points it missed were not a random third — which is the next lesson, and the reason its functionality appears to improve when the roads close.

Part IV

What the monitoring did not see

CHAPTER 7

The round nobody drove

Lesson 7 of 8 · 120 min

2,629 visits were made of 2,904 due. Nord-Ouest reads best of the three districts at 76.4% and its answer is uncertain by thirteen points, against two and a half for the district that visited nearly everything.

7.1 The denominator that was never questioned

Every functionality figure in the last two lessons divided by **visits made**. Nobody chose that denominator; it is what a `groupby` produces from a file whose rows are visits.

The denominator you meant is **visits due**, and the register knows what that is: 242 points, twelve scheduled rounds.

```python
import pandas as pd

points = pd.read_csv("water-point-monitoring-2024.v1.csv", parse_dates=["visit_date"])

due = points["water_point_id"].nunique() * 12
made = len(points)
print(f"due {due:,}, made {made:,}, coverage {made / due:.1%}")
```

```r
library(dplyr)

points |> summarise(
  due = n_distinct(water_point_id) * 12,
  made = n(),
  coverage = n() / (n_distinct(water_point_id) * 12)
)
```

2,629 of 2,904 — 90.5% coverage. Two hundred and seventy-five rounds were never driven, and the register does not contain a row saying so. This is the DHIS2 course's reporting rate arriving in a different file format: **absence is not a value, and nothing in the data will remind you it happened.**

7.2 Coverage is not spread evenly, which is the whole problem

```python
coverage = points.groupby("district").agg(
    points=("water_point_id", "nunique"),
    visits=("water_point_id", "size"),
)
coverage["due"] = coverage["points"] * 12
coverage["coverage"] = coverage["visits"] / coverage["due"]
print(coverage.round(3))
```

```r
points |>
  summarise(points = n_distinct(water_point_id), visits = n(), .by = district) |>
  mutate(due = points * 12, coverage = visits / due)
```

District	Points	Visits made	Due	Coverage
Sud-Est	79	913	948	**96.3%**
Centre	79	882	948	93.0%
Nord-Ouest	84	834	1,008	**82.7%**

Nord-Ouest's rounds collapse when the roads close.

```python
monthly = points.pivot_table(
    index=points["visit_date"].dt.month, columns="district",
    values="water_point_id", aggfunc="size",
)
print(monthly)
```

```r
points |> count(district, month = lubridate::month(visit_date)) |>
  tidyr::pivot_wider(names_from = district, values_from = n)
```

From about 77 points a month to between 51 and 60 in June, July, August and September. Centre and Sud-Est barely move.

7.3 Bound the answer

The honest response to a denominator you cannot fully observe is not a correction factor. It is a **range**, computed from the two assumptions that bracket the truth.

- **Upper bound**: every missed visit would have found a working point. That is the observed rate, which already assumes the missed visits look like the seen ones.
- **Lower bound**: every missed visit would have found a broken point.

```python
WORKING = {"functional", "partially-functional"}
ok = points["functional_status"].isin(WORKING)

bounds = points.assign(ok=ok).groupby("district").agg(
    ok=("ok", "sum"), made=("ok", "size"), points=("water_point_id", "nunique"),
)
bounds["due"] = bounds["points"] * 12
bounds["observed"] = bounds["ok"] / bounds["made"]
bounds["lower"] = bounds["ok"] / bounds["due"]
bounds["band"] = bounds["observed"] - bounds["lower"]
print((bounds[["observed", "lower", "band"]] * 100).round(1))
```

```r
points |>
  mutate(ok = functional_status %in% c("functional", "partially-functional")) |>
  summarise(ok = sum(ok), made = n(), pts = n_distinct(water_point_id),
            .by = district) |>
  mutate(due = pts * 12, observed = ok / made, lower = ok / due)
```

District	Observed	Lower bound	Band
Nord-Ouest	**76.4%**	63.2%	**13.2**
Centre	75.7%	70.5%	5.3
Sud-Est	71.4%	68.8%	2.6

Nord-Ouest is the best district on the observed figure and the only one whose ranking is not safe. Its band overlaps both other districts completely. Sud-Est looks worst and is the only one you can be confident about, because it visited nearly everything.

A number computed on 96% of its denominator and a number computed on 83% are not comparable, and nothing in the table says so unless you put it there.

7.4 Is the missingness random?

The bounds are wide because they assume nothing. Narrowing them means arguing that the missed visits resemble the made ones, and that argument is testable.

```
observed = points.pivot_table(
    index="water_point_id", columns="round", values="functional_status",
    aggfunc="first",
)
followed_by_gap = []
for point, row in observed.iterrows():
    for r in range(1, 12):
        if pd.isna(row.get(r)):
            continue
        broken = row[r] not in WORKING
        followed_by_gap.append((pd.isna(row.get(r + 1)), broken))

check = pd.DataFrame(followed_by_gap, columns=["gap_next", "broken"])
print(check.groupby("gap_next")["broken"].agg(["mean", "size"]).round(3))
```

```
# Same shape: was the point broken at the visit immediately before a missed round?
```

26.7% of visits that were followed by a missed round found a broken point, against 24.7% of visits that were followed by a made one. The difference points the way you would fear — a point known to be broken is a little likelier to be skipped — and it is small enough that it neither proves the bias nor rules it out.

That is a legitimate finding and it should be reported as one. **A test that comes back ambiguous is not a failed test**; it tells you the bounds cannot honestly be narrowed, which is what you needed to know.

7.5 What not to do

Three repairs that all look reasonable.

Carry the last observation forward. Fill each missing round with the point's previous status. It gives 74.2% overall, almost identical to the observed rate, and it is the assumption that a broken point stays broken and a working one keeps working — which is exactly the thing in question. It manufactures precision without adding information.

Impute from the district mean. Assumes the missed points are like the visited ones in the same district, which is the assumption the whole lesson doubts.

Drop the district. Removes the worst-covered district from the report, which is nearly always also the hardest to reach and the worst served.

```
print("Reported: 76.4% (observed), 63.2% to 76.4% (bounds), coverage 82.7%")
print("Not reported: a single imputed number that hides which of these it is")
```

```
# The band and the coverage are the finding. Neither is a caveat.
```

7.6 Report coverage beside every rate

```
Water point functionality by district, 2024

  District      Functionality   Coverage   Range if unvisited were broken
  Sud-Est          71.4%          96.3%    68.8 - 71.4
  Centre           75.7%          93.0%    70.5 - 75.7
  Nord-Ouest       76.4%          82.7%    63.2 - 76.4

  275 of 2,904 scheduled visits were not made, concentrated in Nord-Ouest
  during June to September when roads are impassable. Districts are not
  ranked here: Nord-Ouest's range overlaps both others.
```

The rate, the coverage, the range, and a sentence declining to rank. **Refusing to rank is a result**, and it is more useful than a league table that reverses the moment somebody drives the missing rounds.

7.7 What comes next

You now have every WASH number this course can produce and a set of caveats attached to each. The last lesson is the report they go into — which figures belong on the same page, which denominators must be stated, and the one table that tells a reader what may be compared with what.

CHAPTER 8

Eleven numbers and six denominators

Lesson 8 of 8 · 120 min

Everything this course computed, on one page, with the table that says which figures may be compared with which. Six different denominators across eleven indicators, and no layout will warn a reader about that on its own.

8.1 Count the denominators before writing anything

The course has produced eleven indicators. They come from two files, and they run on six different denominators.

Indicator	Denominator	n
At least basic drinking water	All households	2,403
At least basic sanitation	All households	2,403
Basic hygiene service	All households	2,403
Open defecation	All households	2,403
Below 15 L/person/day	All households	2,403
Round trip over 30 minutes	All households	2,403
E. coli risk class	**Households tested**	802
Chlorine at the household	**Households tested**	961
Water point functionality	**Monitoring visits made**	2,629
Points working at every visit	**Water points**	242
Population-weighted functionality	**People served**	~96,700
Chlorine at the water point	**Visits tested**	811

Six denominators, and only the first six rows share one. Every horizontal comparison across a boundary in that table needs a sentence, and every one made without a sentence will be made wrongly.

```
import pandas as pd

denominators = pd.DataFrame({
    "indicator": ["basic water", "basic sanitation", "basic hygiene",
                  "E. coli safe", "point functionality", "reliable points"],
    "denominator": ["households", "households", "households",
                    "households tested", "visits made", "water points"],
    "n": [2403, 2403, 2403, 802, 2629, 242],
})
print(denominators)
```

```
# Build this table first, in code, and print it above the results.
```

Write it as data, not as prose. A denominator table generated by the same script that computed the numbers cannot drift from them; a paragraph describing the denominators can and will.

8.2 The report

```
WASH situation, three districts, 2024

ACCESS                                          n = 2,403 households
  At least basic drinking water     56.3%       (82.3% on an improved
                                                 source; 25.0 points sit on
                                                 limited service for
                                                 collection time alone)
  At least basic sanitation         43.2%       (62.0% have an improved
                                                 facility; 18.8 points are
                                                 shared)
  Basic hygiene service             34.1%       (67.8% have a facility;
                                                 soap observed at half)
  Open defecation                   13.3%

  Safely managed not computable: availability was not asked and quality was
  tested on a subsample.

QUANTITY AND TIME                               n = 2,403 households
  Median                          23.7 L/person/day
  Below Sphere minimum of 15 L      13.4%       323 households
  Round trip over 30 minutes        39.3%       944 households
  Queue time not separated from walking time. Distance not recorded.

WATER QUALITY                                   n = 802 tested (33.4%)
  Free from E. coli                 53.6%
  High risk (>100 CFU/100 mL)        7.5%
  Improved sources                  61.1% safe  n = 625
  Unimproved and surface            27.1% safe  n = 177

  Tested subsample over-represents untreated water: 533 households reporting
  home treatment have no chlorine reading.

WATER POINT FUNCTIONALITY                       242 points, 2,904 visits due
  Visits finding a working point    74.4%       2,629 visits made (90.5%)
  Points working at every visit     33.9%       82 of 242
  Population-weighted              83.3%       ~96,700 users
  Range if every missed visit found a failure: 67.4% - 74.4%

  Failure shapes: 82 never failed, 80 intermittent, 46 seasonal, 34 down at
  year end. Seasonality is confined to protected wells and springs.

  Median days down at the visit: 68 under community committees, 29 under
  private operators.

NOT COMPARABLE
  Water quality figures are on a tested subsample and do not share the access
  denominator. Functionality figures count visits, points and people and are
  three different quantities. Districts are not ranked: Nord-Ouest visited
  82.7% of its due rounds against Sud-Est's 96.3%.
```

8.3 Four rules the layout has to obey

Group by denominator, not by theme. The block headings above are denominators wearing topic names. A reader who takes one number from each block has taken four numbers with four bases, and the blank line between blocks is what tells them so.

Put n in the heading, not the footnote. A footnote is read after the comparison has already been made.

State what is not computable. Safely managed drinking water, queue time and distance are all absent, and each is something a reader will assume is in a WASH report. An absent indicator that is not named reads as an indicator at zero or as an oversight.

Put the non-comparability in its own block at the end. Not as a caveat under each number, where it becomes noise, and not in a methods annex, where it is not read.

8.4 What each number is for

A report that only classifies is an assessment. A report that also says what to do is a deliverable, and the WASH figures divide cleanly by the decision they inform.

Finding	Decision it informs
25 points on limited water for collection time	Where to site new points
18.8 points on shared sanitation	Whether the programme is building or subsidising
Soap at half the facilities that exist	Hygiene promotion against hardware
46 seasonal points	Dry-season alternative, not rehabilitation
34 down at year end	Spare parts and management, not new construction
68 days median downtime under committees	The maintenance model itself
Nord-Ouest at 82.7% coverage	The monitoring plan before the next report

The last row is the one most reports omit and the one most likely to change next year's numbers. A monitoring system that cannot reach a district in the rains produces a report that cannot describe it, and fixing that is cheaper than any of the others on the list.

8.5 What this course did not cover

Named plainly, because a WASH analyst will be asked for all of them.

- **Menstrual hygiene management** needs questions this survey does not ask — private space, materials, disposal — and the MHM indicators are not derivable from what is here.
- **School and healthcare facility WASH** are separate instruments with their own ladders, and the household ladder does not transfer to an institution.
- **Faecal sludge management** is the second half of safely managed sanitation and needs a containment and emptying survey.
- **Cost per beneficiary and tariff analysis** need financial records; the `fee_collected` field says whether a fee was taken, not how much or where it went.

A report that names its gaps is more credible than one that appears complete, and a gap named is the first line of the next terms of reference.

8.6 What comes next

Module 4 continues in other sectors — food security and the IPC, protection case management, education attendance — and each brings the same three questions in new vocabulary. You have now met them three times: what did I count, over what, and what should change because of it.

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/wash-analysis

Generated July 28, 2026.