74.4%, 33.9%, 83.3%

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.

Pierre Robentz Cassion

Lesson 5 of 8

WASH Analysis — Service is a year, not a day

What this lesson covers

- What a repeat-visit register makes possible
- Rate one: the share of visits that found a working point
- Rate two: the share of points that worked every time
- Rate three: the share of people with a working point
- Which one to report
- The two points counted twice
- What comes next

What a repeat-visit register makes possible — In 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())
```

What a repeat-visit register makes possible — In R

```
library(dplyr)

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

Rate one: the share of visits that found a working point — In Python

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

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

Rate one: the share of visits that found a working point — In R

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

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

- 74.4% — This is what almost every water point report publishes, and it answers *"if I visit a point at random, will it...
- Partially functional counts as working — A point running at reduced yield is providing water, and classifying it as...

Rate two: the share of points that worked every time — In 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()}")
```

Rate two: the share of points that worked every time — In R

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

Rate two: the share of points that worked every time

- 33.9% — 82 of 242 points
- 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

Rate three: the share of people with a working point — In 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%}")
```

Rate three: the share of people with a working point — In R

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

Rate three: the share of people with a working point

- 83.3% — nearly nine points above the visit-level figure

Rate three: the share of people with a working point — In 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))
```

Rate three: the share of people with a working point — In R

```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)
```

Rate three: the share of people with a working point

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

Rate three: the share of people with a working point

- The points that serve the most people break the least — So weighting by population moves the figure up, and the gap...

Which one to report — Example

```
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%.
```

Which one to report

- If you must give one, give the population-weighted rate and say so — because the standard is about people rather than...

The two points counted twice — In Python

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

The two points counted twice — In R

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

What comes next

- Two thirds of these points failed at least once.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/wash-analysis/wash-05-three-functionality-rates