## Tested on a third, censored on a fifth

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.

---

Pierre Robentz Cassion

Lesson 4 of 8

WASH Analysis — What Sphere asks that a ladder does not

# What this lesson covers

- The denominator changes, and the report usually does not
- The JMP risk classes
- Improved is not safe, and this is where you can prove it
- The censored value, and why it is not missing
- What to do with a censored value
- The cross-field inconsistency worth naming
- Report it as its own block
- What comes next

# The denominator changes, and the report usually does not — In 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%}")
```

# The denominator changes, and the report usually does not — In R

```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))
)
```

## The denominator changes, and the report usually does not

- 802 households for E. coli, 961 for chlorine, and 320 for both — The last number is the one that matters: any question. . .
- Print the analysable n for every quality figure — in the table rather than in a footnote

# The JMP risk classes — In 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))
```

# The JMP risk classes — In R

```
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))
```

# The JMP risk classes

| 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% |

# The JMP risk classes

- Report the classes, not a mean — E

# Improved is not safe, and this is where you can prove it — In Python

```
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))
```

# Improved is not safe, and this is where you can prove it — In R

```
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)
```

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

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

# The censored value, and why it is not missing — In Python

```
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())
```

# The censored value, and why it is not missing — In R

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

# The censored value, and why it is not missing — In Python

```
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%}")
```

# The censored value, and why it is not missing — In R

```
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)))
```

## The censored value, and why it is not missing

- 30.9% against 46.5% — Dropping the censored readings raises apparent compliance with the chlorination target by nearly. . .

## What to do with a censored value

- Classify rather than average — The target is a range, so a censored reading answers the question perfectly: `<0.1` is…
- Substitute the limit, or half of it — if you genuinely need a mean
- Report the share below the limit as its own number — 33.4% of these readings are below detection, which is a finding…

# What to do with a censored value — In Python

```
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)
```

# What to do with a censored value — In R

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

## What to do with a censored value

- Never let a detection limit shrink a denominator — It is the single most common way a water quality report becomes...

# The cross-field inconsistency worth naming — In Python

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

# The cross-field inconsistency worth naming — In R

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

# Report it as its own block — Example (cont.)

```
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
```

# Report it as its own block — Example (cont.)

```
chlorine reading, so the tested subsample over-represents untreated water.
```

## What comes next

- Everything so far is a cross-section: what was true on the day someone called.

## Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/wash-analysis/wash-04-water-quality