Missingness that is not an accident

A global completeness figure hides the only thing that matters. Break missingness down by site and week, name the mechanism, and put a number on what dropping the incomplete rows does to your ranking.

Pierre Robentz Cassion

Lesson 2 of 8

Data Cleaning and Validation — Before you change anything

What this lesson covers

- 5.4% is not a finding
- Break it down by group before you do anything else
- Then by time, inside the group that stands out
- Name the mechanism, in words the report can carry
- Put a number on what dropping them costs
- The three responses, and when each is honest
- Missing rows, not just missing values
- What comes next

5.4% is not a finding

- A completeness figure only means something once you know where the holes are — Scattered missingness costs precision

Break it down by group before you do anything else — In Python

```
by_commune = (
    muac.assign(missing_age=muac["age_months"].isna())
    .groupby("commune")
    .agg(missing=("missing_age", "mean"), n=("missing_age", "size"))
    .sort_values("missing", ascending=False)
)
print((by_commune["missing"] * 100).round(1))
```

Break it down by group before you do anything else — In R

```
muac |>
  group_by(commune) |>
  summarise(missing = mean(is.na(age_months)), n = n()) |>
  arrange(desc(missing)) |>
  mutate(missing = round(100 * missing, 1))
```

Break it down by group before you do anything else

Commune	Missing age	Rows
Gros-Morne	17.7%	361
Anse-Rouge	5.7%	229
Saint-Michel	5.1%	335
Dessalines	4.9%	450
Ennery	3.0%	263

Then by time, inside the group that stands out — In Python

```
gros_morne = muac[muac["commune"] == "Gros-Morne"].copy()
gros_morne["week"] = gros_morne["screening_date"].dt.to_period("W").dt.start_time

print(
    gros_morne.groupby("week")["age_months"]
    .apply(lambda s: s.isna().mean())
    .sort_values(ascending=False)
    .head()
)
```

Then by time, inside the group that stands out — In R

```
muac |>
  filter(commune == "Gros-Morne") |>
  mutate(week = lubridate::floor_date(screening_date, "week", week_start = 1)) |>
  group_by(week) |>
  summarise(missing = mean(is.na(age_months)), n = n()) |>
  arrange(desc(missing)) |>
  head()
```

Name the mechanism, in words the report can carry

- **Missing completely at random.** The holes are unrelated to anything, including the value that is missing. Dropping...
- **Missing at random.** The holes depend on something you *observed* — a commune, a week, an enumerator. Dropping them...
- **Missing not at random.** The holes depend on the missing value itself. The sickest children are the ones the queue...

Name the mechanism, in words the report can carry

The category is not a statistical technicality. It is the difference between "we dropped 226 rows" and "we dropped 226 rows, 54 of them from one commune in one week, which moved that commune's rate by 1.1 points".

Put a number on what dropping them costs — In Python

```
def gam_rate(df):
    assessed = df["muac_mm"].notna() | df["oedema"].notna()
    cases = assessed & ((df["muac_mm"] < 125) | (df["oedema"] == True))
    return pd.Series({
        "assessed": int(assessed.sum()),
        "cases": int(cases.sum()),
        "rate": round(cases.sum() / assessed.sum(), 3),
    })


all_rows = muac.groupby("commune").apply(gam_rate)
complete = muac[muac["age_months"].notna()].groupby("commune").apply(gam_rate)

comparison = all_rows.join(complete, rsuffix="_complete")
comparison["shift"] = comparison["rate_complete"] - comparison["rate"]
print(comparison.sort_values("shift"))
```

Put a number on what dropping them costs — In R (cont.)

```r
gam_rate <- function(df) {
  df |>
    mutate(
      assessed = !is.na(muac_mm) | !is.na(oedema),
      case     = assessed & (muac_mm < 125 | oedema)
    ) |>
    summarise(
      assessed = sum(assessed, na.rm = TRUE),
      cases    = sum(case, na.rm = TRUE),
      rate     = round(cases / assessed, 3),
      .by      = commune
    )
}

comparison <- gam_rate(muac) |>
  left_join(gam_rate(filter(muac, !is.na(age_months))),
```

Put a number on what dropping them costs — In R (cont.)

```
          by = "commune", suffix = c("", "_complete")) |>
mutate(shift = rate_complete - rate) |>
arrange(shift)
```

Put a number on what dropping them costs

Commune	All rows	Complete cases	Shift
Gros-Morne	14.5%	13.4%	-1.1 pt
Anse-Rouge	15.6%	16.5%	+0.9 pt
Dessalines	5.8%	5.4%	-0.4 pt
Saint-Michel	7.6%	7.6%	0.0 pt

The three responses, and when each is honest

- Drop them, and report the drop — Legitimate when the missingness is scattered and you have shown it is
- Keep them, in a category of their own — Often the right answer for a categorical variable, because "not recorded" is a...
- Impute — carefully, and rarely — For a variable used to *disaggregate* rather than to compute, filling missing age from...

The three responses, and when each is honest — In Python

```
muac["age_imputed"] = muac["age_months"].isna()
muac["age_months_filled"] = muac.groupby("commune")["age_months"].transform(
    lambda s: s.fillna(s.median())
)
```

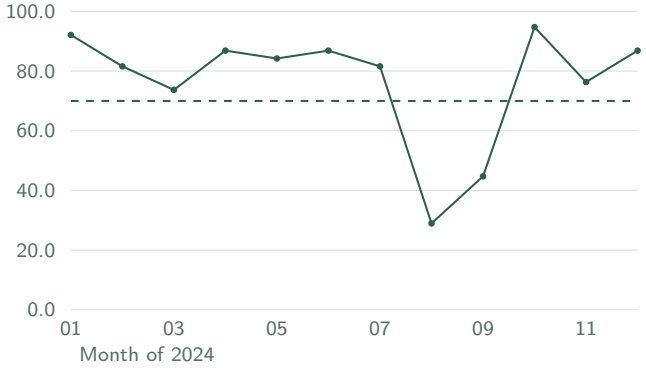
The three responses, and when each is honest — In R

```
muac <- muac |>
  mutate(age_imputed = is.na(age_months)) |>
  group_by(commune) |>
  mutate(age_months_filled = ifelse(is.na(age_months),
                                    median(age_months, na.rm = TRUE),
                                    age_months)) |>
  ungroup()
```

The three responses, and when each is honest

- The flag column is the point — An imputed value that is indistinguishable from a measured one has stopped being an...

Missing rows, not just missing values



Share of health facilities submitting a monthly report through 2024. The collapse across August and September moves together across the district, which is what makes it a reporting event rather than a facility-level failure.

Missing rows, not just missing values — In Python

```
vax["reported"] = vax["report_submitted"] == True

reporting_rate = vax.groupby("period")["reported"].mean()
coverage = (
    vax[vax["reported"]]
    .groupby("period")
    .apply(lambda g: g["doses_administered"].sum() / g["target_population"].sum())
)
print(pd.DataFrame({"reporting_rate": reporting_rate, "coverage": coverage}))
```

Missing rows, not just missing values — In R

```
vax |>
  summarise(
    reporting_rate = mean(report_submitted),
    coverage = sum(doses_administered[report_submitted]) /
               sum(target_population[report_submitted]),
    .by = period
  )
```

What comes next

- Missingness is a hole where something should be.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-cleaning-and-validation/cleaning-02-missingness