Finding what is wrong with the data

Profile a fresh export systematically — missingness by site and week, implausible measurements, duplicates with no clean key, and codes that contradict each other.

Pierre Robentz Cassion

Lesson 5 of 8

Data Analysis Foundations for M&E

What this lesson covers

- Profile before you analyse
- Check 1: completeness, by group and not overall
- Check 2: implausible values
- Check 3: duplicates, including the ones without a clean key
- Check 4: internal contradictions
- Check 5: inconsistent coding of the same thing
- A profile function worth keeping
- What comes next

Profile before you analyse

- You have a tidy table with the right types.

Check 1: completeness, by group and not overall — In Python

```
overall = muac.isna().mean().sort_values(ascending=False)
print(overall)

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

Check 1: completeness, by group and not overall — In R

```
muac |>
  summarise(across(everything(), ~ mean(is.na(.x)))) |>
  tidyr::pivot_longer(everything(), names_to = "column", values_to = "missing") |>
  arrange(desc(missing))

muac |>
  group_by(commune) |>
  summarise(
    missing_age = mean(is.na(age_months)),
    n = n()
  ) |>
  arrange(desc(missing_age))
```

Check 1: completeness, by group and not overall — In Python

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

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

Check 1: completeness, by group and not overall — In R

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

Check 1: completeness, by group and not overall

Missingness that clusters in one site or one week is the difference between losing precision and losing the answer. Always break it down before you decide what to do with it.

Check 2: implausible values — In Python

```
implausible = muac[(muac["muac_mm"] < 80) | (muac["muac_mm"] > 220)]
print(implausible[["child_id", "commune", "age_months", "muac_mm"]])

print(muac["muac_mm"].describe())
```

Check 2: implausible values — In R

```
muac |>
  filter(muac_mm < 80 | muac_mm > 220) |>
  select(child_id, commune, age_months, muac_mm)

summary(muac$muac_mm)
```

Check 2: implausible values — In Python

```
print(muac["age_months"].describe())
print(muac[(muac["age_months"] < 6) | (muac["age_months"] > 59)].shape[0])
```

Check 2: implausible values — In R

```r
summary(muac$age_months)
sum(muac$age_months < 6 | muac$age_months > 59, na.rm = TRUE)
```

Check 3: duplicates, including the ones without a clean key — In Python

```
exact = muac[muac.duplicated(keep=False)]
print(f"{len(exact)} rows in exact-duplicate groups")

repeated_ids = muac[muac.duplicated(subset=["child_id"], keep=False)]
print(f"{len(repeated_ids)} rows sharing a child_id")
```

Check 3: duplicates, including the ones without a clean key — In R

```r
sum(duplicated(muac))

muac |>
  group_by(child_id) |>
  filter(n() > 1) |>
  arrange(child_id)
```

Check 3: duplicates, including the ones without a clean key — In Python

```
suspects = (
    muac.groupby(["commune", "screening_date", "age_months", "sex", "muac_mm"])
    .filter(lambda g: len(g) > 1)
    .sort_values(["commune", "screening_date", "muac_mm"])
)

print(suspects[["child_id", "commune", "screening_date", "age_months", "sex", "muac_mm"]])
```

Check 3: duplicates, including the ones without a clean key — In R

```
muac |>
  group_by(commune, screening_date, age_months, sex, muac_mm) |>
  filter(n() > 1) |>
  arrange(commune, screening_date, muac_mm) |>
  select(child_id, commune, screening_date, age_months, sex, muac_mm)
```

Check 4: internal contradictions — In Python

```
contradiction_a = muac[
    muac["outcome"].isin(["referred-tsfp", "referred-otp", "referred-sc"])
    & muac["muac_mm"].isna()
]
print(len(contradiction_a))
```

Check 4: internal contradictions — In R

```
muac |>
  filter(outcome %in% c("referred-tsfp", "referred-otp", "referred-sc"),
         is.na(muac_mm)) |>
  nrow()
```

Check 4: internal contradictions — In Python

```
def expected_outcome(row):
    if pd.isna(row["muac_mm"]):
        return None
    if row["oedema"] is True or row["muac_mm"] < 115:
        return "referred-otp"
    if row["muac_mm"] < 125:
        return "referred-tsfp"
    return "no-action"

check = muac.assign(expected=muac.apply(expected_outcome, axis=1))
mismatched = check[
    check["expected"].notna()
    & (check["expected"] == "no-action")
    & (check["outcome"] != "no-action")
]
print(len(mismatched))
```

Check 4: internal contradictions — In R

```r
muac |>
  mutate(
    expected = case_when(
      is.na(muac_mm)              ~ NA_character_,
      oedema | muac_mm < 115      ~ "referred-otp",
      muac_mm < 125               ~ "referred-tsfp",
      TRUE                        ~ "no-action"
    )
  ) |>
  filter(!is.na(expected), expected == "no-action", outcome != "no-action") |>
  nrow()
```

Check 5: inconsistent coding of the same thing — In Python

```
raw = pd.read_csv(PATH, dtype={"oedema": "string"})
print(raw.groupby("commune")["oedema"].value_counts(dropna=False))
```

Check 5: inconsistent coding of the same thing — In R

```
read_csv(PATH, col_types = cols(.default = col_character())) |>
  count(commune, oedema)
```

A profile function worth keeping — In Python

```python
def profile(df, group=None):
    report = {
        "rows": len(df),
        "columns": df.shape[1],
        "duplicate_rows": int(df.duplicated().sum()),
        "missing": df.isna().mean().round(4).to_dict(),
    }
    if group:
        report["missing_by_group"] = (
            df.isna().groupby(df[group]).mean().round(4).to_dict("index")
        )
    return report


import json
print(json.dumps(profile(muac, group="commune"), indent=2, default=str))
```

A profile function worth keeping — In R

```r
profile <- function(df, group = NULL) {
  out <- list(
    rows = nrow(df),
    columns = ncol(df),
    duplicate_rows = sum(duplicated(df)),
    missing = sapply(df, function(x) round(mean(is.na(x)), 4))
  )
  if (!is.null(group)) {
    out$missing_by_group <- df |>
      group_by(.data[[group]]) |>
      summarise(across(everything(), ~ round(mean(is.na(.x)), 4)))
  }
  out
}

str(profile(muac, group = "commune"))
```

What comes next

- You now know what is wrong.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-analysis-foundations/foundations-05-data-quality