Cleaning decisions and the cleaning log

Decide what to do about each defect, quantify what the decision costs the final number, and record it so an auditor — or you in six months — can follow the reasoning.

Pierre Robentz Cassion

Lesson 6 of 8

Data Analysis Foundations for M&E

What this lesson covers

- Cleaning is a set of decisions, not a script
- The four options
- Working through this register
- Quantify what the decision costs
- The cleaning log
- The one thing never to do
- What comes next

Cleaning is a set of decisions, not a script

- The profiling in the last lesson produced a list of faults.

The four options

Option	When it applies	What it costs
Correct	You know what the value was meant to be	Nothing, if you are right
Set to missing	The value is wrong and unrecoverable	Precision, and possibly bias
Drop the row	The row is not a real observation	Caseload, if you are wrong
Keep and flag	The value may be right, and you cannot tell	Nothing, but the reader must handle it

Working through this register

- Unit errors — seven records in centimetres — Recoverable

Working through this register — In Python

```
import numpy as np

unit_error = muac["muac_mm"].notna() & (muac["muac_mm"] < 40)
n_unit_error = int(unit_error.sum())

muac.loc[unit_error, "muac_mm"] = muac.loc[unit_error, "muac_mm"] * 10
```

Working through this register — In R

```r
unit_error <- !is.na(muac$muac_mm) & muac$muac_mm < 40
n_unit_error <- sum(unit_error)

muac <- muac |>
  mutate(muac_mm = if_else(!is.na(muac_mm) & muac_mm < 40, muac_mm * 10L, muac_mm))
```

Working through this register

- Implausible values that are not unit errors — Not recoverable

Working through this register — In Python

```
implausible = muac["muac_mm"].notna() & (
    (muac["muac_mm"] < 80) | (muac["muac_mm"] > 220)
)
n_implausible = int(implausible.sum())

muac.loc[implausible, "muac_mm"] = np.nan
```

Working through this register — In R

```r
implausible <- !is.na(muac$muac_mm) & (muac$muac_mm < 80 | muac$muac_mm > 220)
n_implausible <- sum(implausible)

muac <- muac |>
  mutate(muac_mm = if_else(implausible, NA_integer_, muac_mm))
```

Working through this register

- Exact duplicates — twelve rows — Drop

Working through this register — In Python

```
before = len(muac)
muac = muac.drop_duplicates()
n_exact_dupes = before - len(muac)
```

Working through this register — In R

```r
before <- nrow(muac)
muac <- distinct(muac)
n_exact_dupes <- before - nrow(muac)
```

Working through this register

- Re-registrations under a new identifier — six suspected — Keep and flag

Working through this register — In Python

```
key = ["commune", "screening_date", "age_months", "sex", "muac_mm"]
muac["suspected_duplicate"] = muac.duplicated(subset=key, keep=False) & muac[
    "muac_mm"
].notna()

n_suspected = int(muac["suspected_duplicate"].sum())
```

Working through this register — In R

```
muac <- muac |>
  group_by(commune, screening_date, age_months, sex, muac_mm) |>
  mutate(suspected_duplicate = n() > 1 & !is.na(muac_mm)) |>
  ungroup()

n_suspected <- sum(muac$suspected_duplicate)
```

Working through this register

- Inconsistent oedema coding — Recoverable

Working through this register — In Python

```
oedema_map = {
    "true": True, "TRUE": True, "Y": True, "y": True, "yes": True,
    "false": False, "FALSE": False, "N": False, "n": False, "no": False,
}
muac["oedema"] = muac["oedema"].astype("string").str.strip().map(oedema_map)
n_oedema_missing = int(muac["oedema"].isna().sum())
```

Working through this register — In R

```r
muac <- muac |>
  mutate(
    oedema = case_when(
      tolower(trimws(oedema)) %in% c("true", "y", "yes") ~ TRUE,
      tolower(trimws(oedema)) %in% c("false", "n", "no") ~ FALSE,
      TRUE ~ NA
    )
  )

n_oedema_missing <- sum(is.na(muac$oedema))
```

Working through this register

- Missing age clustered in Gros-Morne — This is the one that needs a decision rather than a rule, and it is the subject. . .

Quantify what the decision costs — In Python (cont.)

```
gam_threshold = 125

complete = muac.dropna(subset=["age_months", "muac_mm"])
all_measured = muac.dropna(subset=["muac_mm"])

def gam_rate(df):
    return (df["muac_mm"] < gam_threshold).mean()

comparison = pd.DataFrame(
    {
        "dropping_missing_age": complete.groupby("commune").apply(gam_rate),
        "keeping_missing_age": all_measured.groupby("commune").apply(gam_rate),
    }
)
comparison["difference"] = (
    comparison["dropping_missing_age"] - comparison["keeping_missing_age"]
```

Quantify what the decision costs — In Python (cont.)

```
)
print(comparison.sort_values("difference", ascending=False).round(4))
```

Quantify what the decision costs — In R

```r
gam_rate <- function(df) mean(df$muac_mm < 125, na.rm = TRUE)

complete <- muac |> filter(!is.na(age_months), !is.na(muac_mm))
all_measured <- muac |> filter(!is.na(muac_mm))

comparison <- full_join(
  complete |> group_by(commune) |> summarise(dropping = gam_rate(pick(everything()))),
  all_measured |> group_by(commune) |> summarise(keeping = gam_rate(pick(everything()))),
  by = "commune"
) |>
  mutate(difference = dropping - keeping) |>
  arrange(desc(abs(difference)))

comparison
```

Quantify what the decision costs

The general principle: drop rows for a specific analysis, never from the dataset. A row missing age is still evidence about MUAC. Filtering at the point of use keeps each analysis on the largest sample that supports it.

The cleaning log — In Python (cont.)

```
cleaning_log = pd.DataFrame(
    [
        {
            "rule": "MUAC unit error corrected (cm to mm)",
            "column": "muac_mm",
            "action": "corrected",
            "rows": n_unit_error,
            "rationale": "Values below 40 are centimetres left unconverted. "
            "Plausible mm and cm ranges are disjoint, so the correction is unambiguous.",
        },
        {
            "rule": "Implausible MUAC set to missing",
            "column": "muac_mm",
            "action": "set-missing",
            "rows": n_implausible,
            "rationale": "Outside 80-220 mm is not a measurement for 6-59 months. "
```

The cleaning log — In Python (cont.)

```
        "Row retained; the screening happened.",
    },
    {
        "rule": "Exact duplicate submissions removed",
        "column": "all",
        "action": "dropped",
        "rows": n_exact_dupes,
        "rationale": "Identical rows are one form submitted twice on a poor connection.",
    },
    {
        "rule": "Suspected re-registration flagged",
        "column": "suspected_duplicate",
        "action": "flagged",
        "rows": n_suspected,
        "rationale": "Same commune, date, age, sex and MUAC under different ids. "
        "Cannot be distinguished from coincidence without the paper register.",
```

The cleaning log — In Python (cont.)

```
},
{
    "rule": "Oedema coding harmonised",
    "column": "oedema",
    "action": "corrected",
    "rows": n_oedema_missing,
    "rationale": "Ennery and Desdunes used Y/N in Q1. Blanks remain missing.",
},
{
    "rule": "Missing age retained",
    "column": "age_months",
    "action": "kept",
    "rows": int(muac["age_months"].isna().sum()),
    "rationale": "60% missing in the Gros-Morne week of 10-14 June. Dropping "
    "shifts the commune ranking; MUAC thresholds do not require age.",
},
```

The cleaning log — In Python (cont.)

```
    ]
)

cleaning_log.to_csv("output/tables/cleaning-log.csv", index=False)
print(cleaning_log[["rule", "action", "rows"]])
```

The cleaning log — In R

```
cleaning_log <- tibble::tribble(
  ~rule,                                  ~column,              ~action,       ~rows,
  "MUAC unit error corrected (cm to mm)", "muac_mm",            "corrected",   n_unit_
    error,
  "Implausible MUAC set to missing",      "muac_mm",            "set-missing", n_
    implausible,
  "Exact duplicate submissions removed",  "all",               "dropped",     n_exact_
    dupes,
  "Suspected re-registration flagged",    "suspected_duplicate", "flagged",    n_suspected
    ,
  "Oedema coding harmonised",             "oedema",             "corrected",   n_oedema_
    missing,
  "Missing age retained",                 "age_months",         "kept",        sum(is.na(
    muac$age_months))
)

readr::write_csv(cleaning_log, "output/tables/cleaning-log.csv")
cleaning_log
```

The cleaning log

- **The row counts come from the code**, not from memory. If the rule changes, the count changes with it.
- **It ships with the analysis** — as an annex to the report, not as a file on someone's laptop.
- **The rationale is a sentence, not a category.** "Data quality" is not a rationale.
- **It is written as the cleaning happens**, not reconstructed afterwards. Reconstructed logs are wrong in exactly the...

The one thing never to do

- Do not edit `data/raw/`.

What comes next

- The data is clean and the decisions are recorded.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-analysis-foundations/foundations-06-cleaning-decisions