Rules the sector already gave you

Turn "that looks wrong" into a rule table with a source, a severity and a count. Hard bounds against physical limits, soft bounds against the sector's thresholds, and the discipline of flagging rather than deleting.

Pierre Robentz Cassion

Lesson 5 of 8

Data Cleaning and Validation — Values that cannot be true

What this lesson covers

- "That looks wrong" is not a rule
- The thresholds already exist
- Rules as data, not as if-statements
- Applying the table
- Hard bounds and soft bounds
- Rules that compare two columns
- Flag, do not delete
- The flag rate is itself an indicator
- What comes next

"That looks wrong" is not a rule

- Everyone doing this work develops an eye.

The thresholds already exist

Domain	Rule	Source
MUAC, 6-59 months	Below 80 mm or above 220 mm is not a measurement	WHO / CMAM field practice
Weight-for-height z	Outside -5 to +5 of the reference	WHO 2006 growth standards
SMART anthropometry	Flag z-scores more than 3 SD from the survey mean	SMART plausibility report
Water quantity	Below 15 litres per person per day is below minimum	Sphere
Water collection	Round trip over 30 minutes is limited, not basic, service	JMP service ladders
Free residual chlorine	0.2 to 2.0 mg/L at the point of consumption	Sphere / WHO
Coverage	Doses administered above the target population needs explaining	UNICEF indicator guidance

The thresholds already exist

- Cite the source in the rule — A threshold with a citation survives being questioned; the same number without one gets...

Rules as data, not as if-statements — In Python (cont.)

```
RULES = [
    {
        "id": "muac-range",
        "column": "muac_mm",
        "severity": "error",
        "source": "WHO / CMAM field practice",
        "test": lambda d: d["muac_mm"].between(80, 220) | d["muac_mm"].isna(),
        "message": "MUAC outside 80-220 mm",
    },
    {
        "id": "muac-unit-error",
        "column": "muac_mm",
        "severity": "warning",
        "source": "unit check, cm entered as mm",
        "test": lambda d: ~d["muac_mm"].between(1, 40),
        "message": "MUAC looks like centimetres",
```

Rules as data, not as if-statements — In Python (cont.)

```
    },
    {
        "id": "age-in-scope",
        "column": "age_months",
        "severity": "warning",
        "source": "CMAM screening protocol, 6-59 months",
        "test": lambda d: d["age_months"].between(6, 59) | d["age_months"].isna(),
        "message": "age outside the screening window",
    },
]
```

Rules as data, not as if-statements — In R

```
RULES <- tibble::tribble(
  ~id,               ~column,      ~severity,  ~source,                              ~message,
  "muac-range",      "muac_mm",    "error",    "WHO / CMAM field practice",          "MUAC
     outside 80-220 mm",
  "muac-unit-error", "muac_mm",    "warning",  "unit check, cm entered as mm",       "MUAC
     looks like centimetres",
  "age-in-scope",    "age_months", "warning",  "CMAM screening protocol, 6-59 mo",   "age
     outside the screening window"
)

TESTS <- list(
  `muac-range`      = function(d) dplyr::between(d$muac_mm, 80, 220) | is.na(d$muac_mm),
  `muac-unit-error` = function(d) !dplyr::between(d$muac_mm, 1, 40),
  `age-in-scope`    = function(d) dplyr::between(d$age_months, 6, 59) | is.na(d$age_months)
)
```

Rules as data, not as if-statements

- Every test passes on a missing value — That is deliberate and it catches people out constantly: `NA < 80` is not false,...

Applying the table — In Python (cont.)

```python
def apply_rules(df, rules):
    results = []
    flags = pd.DataFrame(index=df.index)
    for rule in rules:
        ok = rule["test"](df)
        flags[rule["id"]] = ~ok
        results.append({
            "rule": rule["id"],
            "severity": rule["severity"],
            "column": rule["column"],
            "failed": int((~ok).sum()),
            "share": round((~ok).mean(), 4),
            "source": rule["source"],
        })
    return pd.DataFrame(results), flags
```

Applying the table — In Python (cont.)

```
report, flags = apply_rules(muac, RULES)
print(report.to_string(index=False))
```

Applying the table — In R

```r
apply_rules <- function(df, rules, tests) {
  flags <- purrr::map_dfc(rules$id, function(id) {
    tibble::tibble(!!id := !tests[[id]](df))
  })
  report <- rules |>
    dplyr::mutate(
      failed = purrr::map_int(id, ~ sum(flags[[.x]], na.rm = TRUE)),
      share  = round(failed / nrow(df), 4)
    )
  list(report = report, flags = flags)
}

out <- apply_rules(muac, RULES, TESTS)
out$report
```

Applying the table

Rule	Severity	Failed	Share
muac-range	error	7	0.17%
muac-unit-error	warning	7	0.17%
age-in-scope	warning	0	0.00%

Hard bounds and soft bounds

- **Hard bounds** are physically or logically impossible. A negative household size, a MUAC of 450 mm, a discharge date…
- **Soft bounds** are implausible but possible. Ten litres per person per day is below the Sphere minimum, and it is also…

Hard bounds and soft bounds — In Python

```
suspect = wash[wash["litres_per_person_day"] > 60][
    ["household_id", "community", "household_size", "litres_per_person_day", "water_source"]
]
suspect["implied_total"] = suspect["litres_per_person_day"] * suspect["household_size"]
print(suspect)
```

Hard bounds and soft bounds — In R

```
wash |>
  filter(litres_per_person_day > 60) |>
  mutate(implied_total = litres_per_person_day * household_size) |>
  select(household_id, community, household_size, litres_per_person_day,
         water_source, implied_total)
```

Rules that compare two columns — In Python

```
CROSS_RULES = [
    {
        "id": "referral-without-measurement",
        "severity": "warning",
        "test": lambda d: ~(d["outcome"].str.startswith("referred") & d["muac_mm"].isna()),
        "message": "referred but no MUAC recorded",
    },
    {
        "id": "outcome-contradicts-muac",
        "severity": "warning",
        "test": lambda d: ~((d["muac_mm"] >= 125) & (d["oedema"] != True)
                            & (d["outcome"] != "no-action")),
        "message": "no-action expected from the measurement",
    },
]
```

Rules that compare two columns — In R

```
CROSS_TESTS <- list(
  `referral-without-measurement` = function(d)
    !(startsWith(d$outcome, "referred") & is.na(d$muac_mm)),
  `outcome-contradicts-muac` = function(d)
    !(d$muac_mm >= 125 & !d$oedema & d$outcome != "no-action")
)
```

Flag, do not delete — In Python

```
muac = muac.join(flags.add_prefix("flag_"))
muac["flag_any_error"] = flags[[r["id"] for r in RULES if r["severity"] == "error"]].any(axis
    =1)
```

Flag, do not delete — In R

```
muac <- dplyr::bind_cols(muac, dplyr::rename_with(out$flags, ~ paste0("flag_", .x)))
muac$flag_any_error <- dplyr::if_any(dplyr::starts_with("flag_"), identity)
```

Flag, do not delete

- **The count is still available.** "4,218 screenings, 7 excluded for an implausible measurement" needs both numbers, and...
- **The exclusion is reversible.** A reviewer who disagrees with a rule can rerun the analysis without it, in one line.
- **The flags are analysable.** Which brings us to the last section.

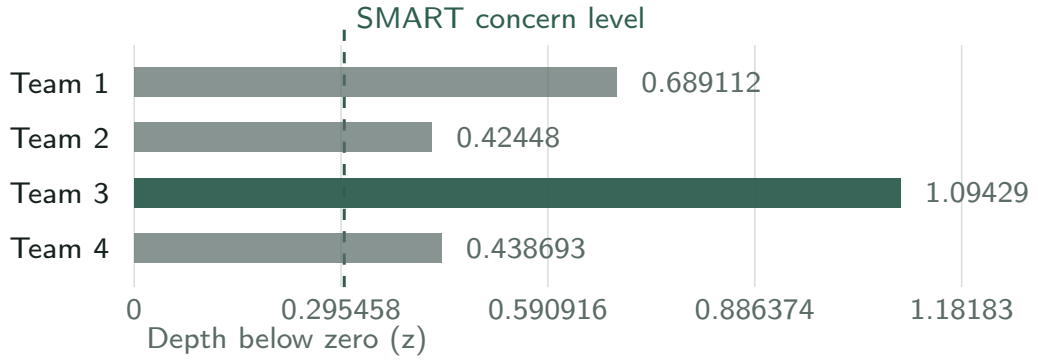
Flag, do not delete — In Python

```
analysis = muac[~muac["flag_any_error"]]
print(f"{len(muac)} screenings, {len(analysis)} analysed, "
      f"{muac['flag_any_error'].sum()} excluded")
```

Flag, do not delete — In R

```
analysis <- muac |> filter(!flag_any_error)
sprintf("%d screenings, %d analysed, %d excluded",
        nrow(muac), nrow(analysis), sum(muac$flag_any_error))
```

The flag rate is itself an indicator



Mean weight-for-height z-score by measurement team in the SMART survey, plotted as depth below zero. Clusters were assigned to teams independently of nutrition status, so a spread of this size between teams is a measurement fault rather than a real difference between populations.

The flag rate is itself an indicator — In Python

```
print(muac.groupby("commune")[["flag_muac_unit_error", "flag_any_error"]].mean())
```

The flag rate is itself an indicator — In R

```r
muac |>
  summarise(across(starts_with("flag_"), mean), .by = commune)
```

The flag rate is itself an indicator

A rule that fires everywhere is a rule about the world. A rule that fires in one place is a rule about a team.

What comes next

- Every rule so far compares a value against a number.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-cleaning-and-validation/cleaning-05-plausibility-rules