Comparing a facility to its own past

Ratio to the rolling median, the small-count problem that makes the biggest outliers meaningless, and the series so stable it is worth a second look.

Pierre Robentz Cassion

Lesson 5 of 8

Data Quality Assessment — Finding the sites worth visiting

What this lesson covers

- Consistency is a comparison, and you get to choose what against
- Ratio to the median
- The small-count problem
- Put an absolute floor beside the ratio
- The direction is not symmetric
- Series that are too stable
- Rank facilities, do not flag rows
- What comes next

Consistency is a comparison, and you get to choose what against

- Against its peers — Facility A reported 40 doses; the district median is 25
- Against its own past — Facility A reported 40 doses this month, against its own median of 25 for the year

Ratio to the median — In Python

```
import pandas as pd

reported = vax[vax["reported"]].copy()

reported["own_median"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"
].transform("median")
reported["ratio"] = reported["doses_administered"] / reported["own_median"]

print(reported.nlargest(5, "ratio")[
    ["facility_id", "antigen", "period", "doses_administered", "own_median", "ratio"]
])
```

Ratio to the median — In R

```
library(dplyr)

reported <- vax |>
  filter(report_submitted) |>
  mutate(own_median = median(doses_administered),
         ratio = doses_administered / own_median,
         .by = c(facility_id, antigen))

reported |> slice_max(ratio, n = 5)
```

Ratio to the median

- Median, not mean — The mean is dragged by the very outlier you are looking for, so a value can inflate its own...

Ratio to the median — In Python

```
reported = reported.sort_values(["facility_id", "antigen", "period"])
reported["rolling_median"] = (
    reported.groupby(["facility_id", "antigen"])["doses_administered"]
    .transform(lambda s: s.rolling(6, min_periods=3, center=True).median())
)
```

Ratio to the median — In R

```
reported <- reported |>
  arrange(facility_id, antigen, period) |>
  mutate(rolling_median = zoo::rollapply(doses_administered, 6, median,
                                         partial = TRUE, align = "center"),
         .by = c(facility_id, antigen))
```

The small-count problem

Facility	Antigen	Month	Value	Own median	Ratio
FAC013	mcv2	December	5	3	1.67
FAC037	opv3	September	13	8	1.62
FAC030	mcv2	October	8	5	1.60

Put an absolute floor beside the ratio — In Python

```
RATIO_HIGH, RATIO_LOW, MIN_ABSOLUTE = 2.0, 0.5, 10

reported["difference"] = reported["doses_administered"] - reported["own_median"]
reported["flag"] = (
    ((reported["ratio"] > RATIO_HIGH) | (reported["ratio"] < RATIO_LOW))
    & (reported["difference"].abs() >= MIN_ABSOLUTE)
)
print(f"{reported['flag'].sum()} flags from {len(reported)} facility-antigen-months")
```

Put an absolute floor beside the ratio — In R

```
reported <- reported |>
  mutate(
    difference = doses_administered - own_median,
    flag = (ratio > 2 | ratio < 0.5) & abs(difference) >= 10
  )
```

Put an absolute floor beside the ratio

- That is the finding, and it is one line long — A check that produces one investigable item from 2,094 observations is. . .

The direction is not symmetric

- **A drop** is usually real or reporting. Stock-out, staff absence, a strike, a facility that started reporting to a...
- **A spike** is usually a campaign, an outreach round, or catch-up after a stock-out — all legitimate — or double...

The direction is not symmetric — In Python

```
reported["previous"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"].shift(1)
reported["next"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"].shift(-1)

spike_and_dip = reported[
    (reported["ratio"] > 1.5)
    & (reported["previous"] < reported["own_median"] * 0.7)
]
```

The direction is not symmetric — In R

```
reported <- reported |>
  mutate(previous = lag(doses_administered),
         next_value = lead(doses_administered),
         .by = c(facility_id, antigen))
```

Series that are too stable — In Python

```
stability = (
    reported.groupby(["facility_id", "antigen"])["doses_administered"]
    .agg(["mean", "std", "size"])
)
stability["cv"] = stability["std"] / stability["mean"]
print(stability[stability["size"] >= 8].nsmallest(5, "cv"))
```

Series that are too stable — In R

```
reported |>
  summarise(mean = mean(doses_administered),
            cv = sd(doses_administered) / mean(doses_administered),
            n = n(),
            .by = c(facility_id, antigen)) |>
  filter(n >= 8) |>
  slice_min(cv, n = 5)
```

Rank facilities, do not flag rows — In Python

```
risk = (
    reported.groupby("facility_id")
    .agg(flags=("flag", "sum"), months=("flag", "size"))
    .assign(flag_rate=lambda d: d["flags"] / d["months"])
    .sort_values("flag_rate", ascending=False)
)
print(risk.head(6))
```

Rank facilities, do not flag rows — In R

```
risk <- reported |>
  summarise(flags = sum(flag), months = n(), .by = facility_id) |>
  mutate(flag_rate = flags / months) |>
  arrange(desc(flag_rate))
```

What comes next

- Trend checks compare a number to its own past.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/data-quality-assessment/dqa-05-trend-and-outlier-checks