Quand le registre et le rapport divergent

Recalculer l'indicateur depuis le registre ligne à ligne, le poser à côté de ce qui a été remonté, et décomposer l'écart par unité de rapportage au lieu de le noyer dans une moyenne.

Pierre Robentz Cassion

Leçon 7 sur 8

Jointures et restructuration des données de programme — La table que vous analysez

Ce que couvre cette leçon

- Deux chiffres pour la même chose
- Recalculer depuis le registre
- La décision de codage qui déplace un dénominateur
- Mettez les deux sources dans une seule table
- Décomposez, ne moyennez pas
- Pourquoi deux sources divergent
- Rapportez l'écart, ne le faites pas disparaître
- La suite

Deux chiffres pour la même chose

- Quelque part se trouve un registre ligne à ligne — marques de présence journalières, cahier de dépistage, registre de patients.

Recalculer depuis le registre — En Python

```
attendance["marked"] = attendance["present"].isin(["true", "false"])
attendance["is_present"] = attendance["present"] == "true"

from_register = (
    attendance.merge(roster[["student_id", "school_id"]], on="student_id",
                     how="left", validate="many_to_one")
    .groupby("school_id")
    .agg(marks=("marked", "sum"), present=("is_present", "sum"))
)
from_register["rate"] = from_register["present"] / from_register["marks"]
```

Recalculer depuis le registre — En R

```r
from_register <- attendance |>
  left_join(select(roster, student_id, school_id), by = "student_id",
            relationship = "many-to-one") |>
  summarise(
    marks   = sum(present %in% c("true", "false")),
    present = sum(present == "true", na.rm = TRUE),
    .by = school_id
  ) |>
  mutate(rate = present / marks)
```

La décision de codage qui déplace un dénominateur — En Python

```
mapping = {"true": True, "false": False, "Y": True, "N": False}
attendance["is_present_mapped"] = attendance["present"].map(mapping)

both_ways = (
    attendance.merge(roster[["student_id", "school_id"]], on="student_id")
    .groupby("school_id")
    .agg(
        marks_strict=("is_present", lambda s: s.notna().sum()),
        rate_strict=("is_present", "mean"),
        marks_mapped=("is_present_mapped", lambda s: s.notna().sum()),
        rate_mapped=("is_present_mapped", "mean"),
    )
)
print(both_ways.loc["SCH09"])
```

La décision de codage qui déplace un dénominateur — En R

```r
both_ways <- attendance |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  mutate(mapped = dplyr::recode(present, "Y" = "true", "N" = "false")) |>
  summarise(
    marks_strict = sum(present %in% c("true", "false")),
    rate_strict  = mean(present[present %in% c("true", "false")] == "true"),
    marks_mapped = sum(mapped %in% c("true", "false")),
    rate_mapped  = mean(mapped[mapped %in% c("true", "false")] == "true"),
    .by = school_id
  )
```

La décision de codage qui déplace un dénominateur

SCH09	Strict	Y/N recodés
Marques au dénominateur	2 015	2 865
Taux de présence	86,05 %	85,72 %

Mettez les deux sources dans une seule table — En Python

```
comparison = (
    from_register.rename(columns={"present": "register_present"})
    .join(reported.rename(columns={"present": "reported_present"}), how="outer")
)
comparison["difference"] = comparison["register_present"] - comparison["reported_present"]
comparison["verification_factor"] = (
    comparison["register_present"] / comparison["reported_present"]
)
print(comparison.sort_values("difference").head(10))
```

Mettez les deux sources dans une seule table — En R

```r
comparison <- from_register |>
  full_join(reported, by = c("school_id", "month"),
            suffix = c("_register", "_reported")) |>
  mutate(
    difference = present_register - present_reported,
    verification_factor = present_register / present_reported
  ) |>
  arrange(difference)
```

Décomposez, ne moyennez pas — En Python

```
print(comparison["verification_factor"].describe())
print(comparison[comparison["verification_factor"].sub(1).abs() > 0.05])
```

Décomposez, ne moyennez pas — En R

```
comparison |>
  summarise(across(verification_factor, list(min = min, median = median, max = max)))

comparison |> filter(abs(verification_factor - 1) > 0.05)
```

Décomposez, ne moyennez pas

- Rapportez la distribution et les valeurs extrêmes, jamais la moyenne seule — Une bande de tolérance — tout ce qui sort...

Pourquoi deux sources divergent

- **Des définitions différentes.** Le registre compte des marques ; le rapport compte des enfants inscrits. De loin la…
- **Des périodes différentes.** Le rapport couvre un mois calendaire, le registre a été extrait le 28. Vérifiez les…
- **Des couvertures différentes.** Le rapport inclut une école que l'extraction du registre excluait, ou l'inverse….
- **La transcription.** Quelqu'un a additionné la colonne à la main. C'est la cause que tout le monde suppose et la moins…

Rapportez l'écart, ne le faites pas disparaître — En Python

```
summary = pd.DataFrame({
    "source": ["Line-level register", "Reported monthly returns"],
    "attendance days recorded": [69101, 68240],
    "attendance rate": ["88.5%", "89.1%"],
})
```

Rapportez l'écart, ne le faites pas disparaître — En R

```r
summary <- tibble::tibble(
  source = c("Line-level register", "Reported monthly returns"),
  days   = c(69101, 68240),
  rate   = c("88.5%", "89.1%")
)
```

Rapportez l'écart, ne le faites pas disparaître

Un écart que vous rapportez est un constat. Un écart que vous refermez en silence est un chiffre que personne ne peut reproduire, vous compris, dans six mois.

La suite

- Toutes les pièces sont en place — des jointures démontrables, une granularité énoncée, la bonne forme, un dénominateur, un calendrier complet et une source rapprochée.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/joining-and-reshaping/joining-07-register-vs-aggregate