Une ligne par quoi ?

La granularité d'une table décide du sens de tout nombre qu'on en tire. Ménages contre personnes, épisodes contre patients, et le plusieurs-à-plusieurs qui transforme 2 403 lignes en 13 004 sans que personne l'ait décidé.

Pierre Robentz Cassion

Leçon 3 sur 8

Ce que couvre cette leçon

- Énoncez la granularité à voix haute
- La granularité décide du sens du nombre
- Développer les ménages en personnes
- Pondérer vaut généralement mieux que développer
- Agrégez à la granularité avant de joindre
- Le plusieurs-à-plusieurs que personne n'a choisi
- Des épisodes, pas des personnes
- Inscrivez la granularité dans le nom du fichier
- La suite

Énoncez la granularité à voix haute — En Python

```python
# grain: one household interview
wash = pd.read_csv("wash-household-survey-2024.v1.csv")

# grain: one student per school day
attendance = pd.read_csv("school-attendance-2024.v1.csv")

# grain: one facility x month x antigen
vax = pd.read_csv("vaccination-coverage-2024.v1.csv")
```

Énoncez la granularité à voix haute — En R

```r
# grain: one household interview
wash <- read_csv("wash-household-survey-2024.v1.csv")

# grain: one student per school day
attendance <- read_csv("school-attendance-2024.v1.csv")

# grain: one facility x month x antigen
vax <- read_csv("vaccination-coverage-2024.v1.csv")
```

La granularité décide du sens du nombre — En Python

```
sized = wash[wash["household_size"].notna() & wash["litres_per_person_day"].notna()]

by_household = (sized["litres_per_person_day"] < 15).mean()
by_person = (
    sized.loc[sized["litres_per_person_day"] < 15, "household_size"].sum()
    / sized["household_size"].sum()
)

print(f"households below 15 L/p/d: {by_household:.1%}")
print(f"people in those households: {by_person:.1%}")
```

La granularité décide du sens du nombre — En R

```r
sized <- wash |> filter(!is.na(household_size), !is.na(litres_per_person_day))

sized |>
  summarise(
    by_household = mean(litres_per_person_day < 15),
    by_person = sum(household_size[litres_per_person_day < 15]) / sum(household_size)
  )
```

La granularité décide du sens du nombre

- Indiquez la granularité dans le libellé — `share_of_households_below_15l` et `share_of_people_below_15l` ne peuvent pas...

Développer les ménages en personnes — En Python

```
people = wash.loc[wash["household_size"].notna()].copy()
people["household_size"] = people["household_size"].astype(int)
people = people.loc[people.index.repeat(people["household_size"])]
people["person_index"] = people.groupby("household_id").cumcount() + 1

print(len(wash), "households ->", len(people), "people")
```

Développer les ménages en personnes — En R

```r
people <- wash |>
  filter(!is.na(household_size)) |>
  tidyr::uncount(household_size, .remove = FALSE, .id = "person_index")

cat(nrow(wash), "households ->", nrow(people), "people\n")
```

Développer les ménages en personnes

- **C'est une affirmation, pas une transformation.** Chaque personne du ménage est désormais supposée avoir l'accès à…
- **Rien des individus n'est réel.** `person_index` est une position, pas une personne. Ne désagrégez pas dessus, et ne…
- **Les quarante-six ménages sans taille enregistrée disparaissent.** Dites-le, ou la population développée est…

Pondérer vaut généralement mieux que développer — En Python

```
sized["weight"] = sized["household_size"]

weighted_share = (
    (sized["litres_per_person_day"] < 15) * sized["weight"]
).sum() / sized["weight"].sum()
```

Pondérer vaut généralement mieux que développer — En R

```
sized |>
  summarise(share = weighted.mean(litres_per_person_day < 15, w = household_size))
```

Agrégez à la granularité avant de joindre — En Python

```python
per_student = (
    attendance.assign(present=attendance["present"].map({"true": True, "false": False}))
    .groupby("student_id")
    .agg(days_marked=("present", "size"),
         days_present=("present", "sum"))
    .reset_index()
)
per_student["attendance_rate"] = per_student["days_present"] / per_student["days_marked"]

# grain: one student. Now the join is one-to-one.
summary = roster.merge(per_student, on="student_id", how="left", validate="one_to_one")
```

Agrégez à la granularité avant de joindre — En R

```r
per_student <- attendance |>
  summarise(
    days_marked  = sum(present %in% c(TRUE, FALSE)),
    days_present = sum(present %in% TRUE),
    .by = student_id
  ) |>
  mutate(attendance_rate = days_present / days_marked)

summary <- roster |>
  left_join(per_student, by = "student_id", relationship = "one-to-one")
```

Le plusieurs-à-plusieurs que personne n'a choisi — En Python

```python
# household file: one row per household. member file: one row per person.
# Both carry `community`. Joining on it instead of household_id:
bad = households.merge(members, on="community", how="inner")
```

Le plusieurs-à-plusieurs que personne n'a choisi — En R

```r
bad <- households |> inner_join(members, by = "community")
```

Des épisodes, pas des personnes

- **La charge de cas** est un décompte d'épisodes. Deux admissions du même enfant sont deux admissions, et les rapporter…
- **Les enfants atteints** est un décompte d'enfants distincts, plus petit.
- **Le taux de guérison** a des épisodes au numérateur et au dénominateur, et mélanger un numérateur d'épisodes avec un…

Des épisodes, pas des personnes — En Python

```
print("episodes:", len(admissions))
print("children:", admissions["child_id"].nunique())
print("readmitted:", (admissions.groupby("child_id").size() > 1).sum())
```

Des épisodes, pas des personnes — En R

```r
c(episodes = nrow(admissions),
  children = n_distinct(admissions$child_id),
  readmitted = sum(count(admissions, child_id)$n > 1))
```

Inscrivez la granularité dans le nom du fichier — Exemple

```
outputs/tables/attendance_by_student.csv
outputs/tables/attendance_by_school_month.csv
outputs/tables/coverage_by_facility_period_antigen.csv
```

La suite

- La granularité répond à la question de ce qu'est une ligne.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/joining-and-reshaping/joining-03-grain-and-rosters