Regrouper jusqu'à un numérateur et un dénominateur

groupby, agrégation nommée, et les deux réglages par défaut qui changent silencieusement un dénominateur — groupes manquants écartés et catégories non observées. Produire les deux moitiés d'un indicateur en une seule opération pour qu'elles ne puissent pas diverger.

Pierre Robentz Cassion

Leçon 6 sur 8

Python pour les données de programme — Travailler le tableau

Ce que couvre cette leçon

- Un indicateur, ce sont deux nombres
- L'agrégation nommée
- Deux réglages par défaut qui changent un dénominateur
- Regrouper sur plusieurs clés
- transform : une statistique de groupe sur chaque ligne
- apply, et pourquoi l'éviter
- Les tableaux croisés se lisent mieux dans un rapport
- Écrire le résultat
- Ce qui vient ensuite

Un indicateur, ce sont deux nombres — En Python (suite)

```
import pandas as pd

muac = read_register(RAW / "muac-screening-artibonite-2024.v1.csv")

measured = muac["muac_mm"].notna()
gam = measured & (muac["muac_mm"] < 125)

by_commune = (
    muac.assign(measured=measured, gam=gam)
    .groupby("commune")
    .agg(
        screened=("child_id", "size"),
        measured=("measured", "sum"),
        gam_cases=("gam", "sum"),
    )
)
```

Un indicateur, ce sont deux nombres — En Python (suite)

```
by_commune["gam_rate"] = (by_commune["gam_cases"] / by_commune["measured"]).round(3)
```

Un indicateur, ce sont deux nombres — Exemple

commune	screened	measured	gam_cases	gam_rate
Gros-Morne	361	351	48	0.137
Anse-Rouge	229	223	30	0.135
L-Estere	189	187	18	0.096
Terre-Neuve	241	235	21	0.089

L'agrégation nommée — En Python

```
.agg(
    nom_de_sortie=("colonne_d_entree", "fonction"),
    ...
)
```

L'agrégation nommée — En Python

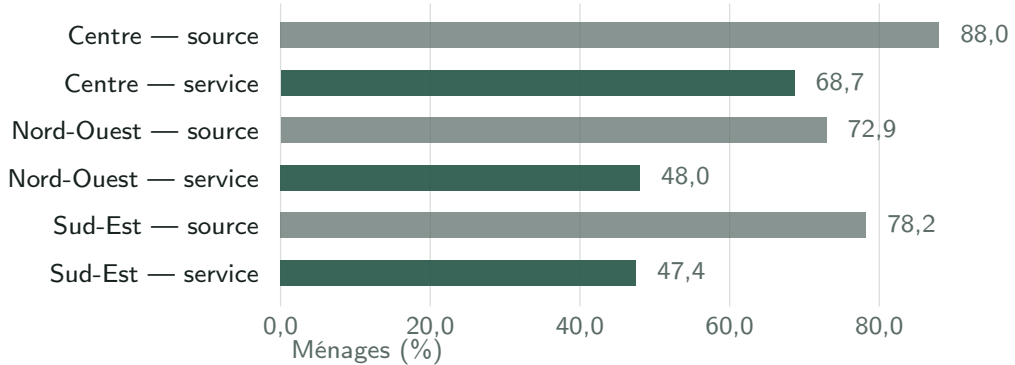
```
muac.groupby("commune")["muac_mm"].mean()                # une colonne, une fonction
muac.groupby("commune").agg({"muac_mm": ["mean", "std"]})  # colonnes MultiIndex
```

L'agrégation nommée — En Python

```
def share_below(series, threshold=125):
    valid = series.dropna()
    return (valid < threshold).mean() if len(valid) else float("nan")

muac.groupby("commune").agg(
    gam_rate=("muac_mm", share_below),
    n=("muac_mm", "count"),
)
```

Deux réglages par défaut qui changent un dénominateur



Source d'eau améliorée contre service de base, par district. Compter les sources surestime la couverture partout ; l'écart correspond aux ménages dont la source est améliorée et située à plus de trente minutes.

Deux réglages par défaut qui changent un dénominateur

- Les clés de groupe manquantes sont écartées

Deux réglages par défaut qui changent un dénominateur — En Python

```python
d = pd.DataFrame({"g": ["a", None, "a"], "v": [1, 2, 3]})

d.groupby("g")["v"].sum()                  # {'a': 4}
d.groupby("g", dropna=False)["v"].sum()    # {'a': 4, nan: 2}
```

Deux réglages par défaut qui changent un dénominateur — En Python

```
assert muac["commune"].notna().all(), "des lignes sans commune seraient ecartees"
```

Deux réglages par défaut qui changent un dénominateur

- Les catégories non observées réapparaissent en lignes vides

Deux réglages par défaut qui changent un dénominateur — En Python

```
c = pd.DataFrame({"k": pd.Categorical(["x", "y"], categories=["x", "y", "z"]),
                  "v": [1, 2]})

len(c.groupby("k", observed=True)["v"].sum())    # 2
len(c.groupby("k", observed=False)["v"].sum())   # 3 -- inclut un "z" vide
```

Deux réglages par défaut qui changent un dénominateur

- Passez-le explicitement — Le comportement par défaut a changé d'une version de pandas à l'autre, et un script qui s'y...

Regrouper sur plusieurs clés — En Python

```
by_commune_month = (
    muac.assign(month=muac["screening_date"].dt.to_period("M"), gam=gam)
    .groupby(["commune", "month"], observed=True)
    .agg(measured=("muac_mm", "count"), gam_cases=("gam", "sum"))
)
```

Regrouper sur plusieurs clés — En Python

```
by_commune_month.loc["Gonaives"]          # une commune, tous les mois
by_commune_month.reset_index()            # retour a des colonnes plates
```

Regrouper sur plusieurs clés — En Python

```
wide = by_commune_month["gam_cases"].unstack("month", fill_value=0)
```

transform : une statistique de groupe sur chaque ligne — En Python

```
muac["commune_mean"] = muac.groupby("commune")["muac_mm"].transform("mean")
muac["vs_commune"] = muac["muac_mm"] - muac["commune_mean"]
```

apply, et pourquoi l'éviter — En Python

```python
# Fonctionne, mais lent, et renvoie un objet dont la forme depend de la fonction.
muac.groupby("commune").apply(lambda g: g["muac_mm"].mean(), include_groups=False)
```

Les tableaux croisés se lisent mieux dans un rapport — En Python

```python
pd.crosstab(muac["commune"], muac["outcome"])
```

```python
pd.crosstab(
    muac["commune"], muac["outcome"],
    values=muac["muac_mm"], aggfunc="mean",
).round(1)
```

Écrire le résultat — En Python

```
OUTPUTS = PROJECT / "outputs" / "tables"
OUTPUTS.mkdir(parents=True, exist_ok=True)

by_commune.reset_index().to_csv(OUTPUTS / "gam_by_commune.csv", index=False)
```

Écrire le résultat — En Python

```
definitions = pd.DataFrame([
    ("MAG", "PB < 125 mm ou oedemes bilateraux prenant le godet",
     "enfants avec une mesure de PB ou une evaluation d'oedemes saisie"),
], columns=["indicateur", "numerateur", "denominateur"])

definitions.to_csv(OUTPUTS / "definitions.csv", index=False)
```

Ce qui vient ensuite

- Le tableau agrège correctement et les deux moitiés de chaque taux proviennent d'une seule opération.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/python-for-programme-data/python-06-grouping-and-aggregation