Des données ordonnées au tableau d'indicateurs

Transformer un jeu de données propre en tableau d'indicateurs désagrégé conforme au cadre logique, avec numérateur, dénominateur et intervalle de confiance explicitement énoncés.

Pierre Robentz Cassion

Leçon 7 sur 8

Fondamentaux de l'analyse de données pour le suivi-évaluation

Ce que couvre cette leçon

- Un indicateur est un numérateur rapporté à un dénominateur, désagrégé
- La fiche de référence d'indicateur
- Construisez le numérateur comme une colonne
- Le tableau d'indicateurs
- La désagrégation
- Énoncez votre degré d'incertitude
- Mettez-le en forme pour le rapport
- La suite

Un indicateur est un numérateur rapporté à un dénominateur, désagrégé

- Cette phrase résume toute la leçon.

La fiche de référence d'indicateur

Champ	Pour notre exemple
Indicateur	Prévalence de la malnutrition aiguë globale (MAG) par PB
Numérateur	Enfants de 6 à 59 mois dépistés avec un PB inférieur à 125 mm, ou porteurs d'œdèmes bilatéraux prenant le godet
Dénominateur	Enfants de 6 à 59 mois disposant d'une mesure de PB valide ou d'une évaluation des œdèmes consignée
Désagrégation	Commune, sexe, tranche d'âge (6-23, 24-59 mois)
Fréquence	Trimestrielle, et annuelle pour le rapport bailleur
Source	Registre de dépistage de masse communautaire, CommCare
Décision éclairée	Quelles communes reçoivent un site PCIMA supplémentaire au trimestre suivant
Valeur de référence	Une MAG égale ou supérieure à 15 % constitue le seuil d'urgence

La fiche de référence d'indicateur

Si vous ne savez pas énoncer le dénominateur en une phrase, vous n'avez pas encore d'indicateur. Vous avez une colonne dont vous vous apprêtez à prendre la moyenne.

Construisez le numérateur comme une colonne — En Python (suite)

```
import numpy as np

MAS_MM = 115
MAG_MM = 125

muac["evalue"] = muac["muac_mm"].notna() | muac["oedema"].notna()

muac["mas"] = np.where(
    ~muac["evalue"],
    np.nan,
    ((muac["muac_mm"] < MAS_MM) | (muac["oedema"] == True)).astype(float),
)

muac["mag"] = np.where(
    ~muac["evalue"],
    np.nan,
```

Construisez le numérateur comme une colonne — En Python (suite)

```
    ((muac["muac_mm"] < MAG_MM) | (muac["oedema"] == True)).astype(float),
)
```

Construisez le numérateur comme une colonne — En R

```r
MAS_MM <- 115
MAG_MM <- 125

muac <- muac |>
  mutate(
    evalue = !is.na(muac_mm) | !is.na(oedema),
    mas = if_else(evalue, (muac_mm < MAS_MM) | oedema %in% TRUE, NA),
    mag = if_else(evalue, (muac_mm < MAG_MM) | oedema %in% TRUE, NA)
  )
```

Construisez le numérateur comme une colonne

- **Les œdèmes signent une MAS indépendamment de la mesure.** Un enfant à 130 mm porteur d'œdèmes bilatéraux prenant le…
- `oedema == True` **plutôt qu'un test de véracité.** Un œdème manquant n'est pas un œdème absent. En R, ‘oedema %in%…

Le tableau d'indicateurs — En Python (suite)

```python
def tableau_indicateurs(df, par):
    groupes = df.groupby(par, dropna=False)
    tableau = groupes.agg(
        depistes=("child_id", "size"),
        denominateur=("evalue", "sum"),
        cas_mas=("mas", "sum"),
        cas_mag=("mag", "sum"),
    )
    tableau["taux_mag"] = tableau["cas_mag"] / tableau["denominateur"]
    tableau["taux_mas"] = tableau["cas_mas"] / tableau["denominateur"]
    return tableau.reset_index()


par_commune = tableau_indicateurs(muac, "commune").sort_values(
    "taux_mag", ascending=False
)
```

Le tableau d'indicateurs — En Python (suite)

```
print(par_commune.round(4))
```

Le tableau d'indicateurs — En R (suite)

```r
tableau_indicateurs <- function(df, par) {
  df |>
    group_by(across(all_of(par))) |>
    summarise(
      depistes     = n(),
      denominateur = sum(evalue),
      cas_mas      = sum(mas, na.rm = TRUE),
      cas_mag      = sum(mag, na.rm = TRUE),
      .groups = "drop"
    ) |>
    mutate(
      taux_mag = cas_mag / denominateur,
      taux_mas = cas_mas / denominateur
    )
}
```

Le tableau d'indicateurs — En R (suite)

```
par_commune <- tableau_indicateurs(muac, "commune") |> arrange(desc(taux_mag))
par_commune
```

La désagrégation — En Python

```
muac["tranche_age"] = pd.cut(
    muac["age_months"],
    bins=[6, 24, 60],
    labels=["6-23 mois", "24-59 mois"],
    right=False,
)

par_sexe = tableau_indicateurs(muac, ["commune", "sex"])
par_age = tableau_indicateurs(muac, "tranche_age")

print(par_age.round(4))
```

La désagrégation — En R

```r
muac <- muac |>
  mutate(
    tranche_age = cut(
      age_months,
      breaks = c(6, 24, 60),
      labels = c("6-23 mois", "24-59 mois"),
      right = FALSE
    )
  )

par_sexe <- tableau_indicateurs(muac, c("commune", "sex"))
par_age <- tableau_indicateurs(muac, "tranche_age")

par_age
```

Énoncez votre degré d'incertitude — En Python (suite)

```python
from scipy.stats import beta

def intervalle(succes, n, confiance=0.95):
    if n == 0:
        return (np.nan, np.nan)
    bas = beta.ppf((1 - confiance) / 2, succes, n - succes + 1) if succes > 0 else 0.0
    haut = beta.ppf(1 - (1 - confiance) / 2, succes + 1, n - succes) if succes < n else 1.0
    return (bas, haut)


bornes = par_commune.apply(
    lambda r: intervalle(r["cas_mag"], r["denominateur"]), axis=1
)
par_commune["mag_bas"] = [b[0] for b in bornes]
par_commune["mag_haut"] = [b[1] for b in bornes]
```

Énoncez votre degré d'incertitude — En Python (suite)

```
print(par_commune[["commune", "denominateur", "taux_mag", "mag_bas", "mag_haut"]].round(4))
```

Énoncez votre degré d'incertitude — En R

```r
intervalle <- function(cas, n) {
  if (n == 0) return(c(NA_real_, NA_real_))
  test <- binom.test(cas, n)
  test$conf.int
}

par_commune <- par_commune |>
  rowwise() |>
  mutate(
    mag_bas  = intervalle(cas_mag, denominateur)[1],
    mag_haut = intervalle(cas_mag, denominateur)[2]
  ) |>
  ungroup()

par_commune |> select(commune, denominateur, taux_mag, mag_bas, mag_haut)
```

Mettez-le en forme pour le rapport — En Python

```
rapport = par_commune.assign(
    mag=lambda d: (d["taux_mag"] * 100).round(1).astype(str) + "%",
    ic=lambda d: "("
    + (d["mag_bas"] * 100).round(1).astype(str)
    + " – "
    + (d["mag_haut"] * 100).round(1).astype(str)
    + ")",
)[["commune", "depistes", "denominateur", "cas_mag", "mag", "ic"]]

rapport.columns = [
    "Commune", "Dépistés", "Évalués", "Cas MAG", "Taux MAG", "IC 95%",
]
rapport.to_csv("output/tables/mag-par-commune.csv", index=False)
print(rapport.to_string(index=False))
```

Mettez-le en forme pour le rapport — En R

```r
rapport <- par_commune |>
  transmute(
    Commune     = commune,
    `Dépistés`  = depistes,
    `Évalués`   = denominateur,
    `Cas MAG`   = cas_mag,
    `Taux MAG`  = sprintf("%.1f%%", taux_mag * 100),
    `IC 95%`    = sprintf("(%.1f - %.1f)", mag_bas * 100, mag_haut * 100)
  )

readr::write_csv(rapport, "output/tables/mag-par-commune.csv")
rapport
```

La suite

- La dernière leçon fait en sorte que tout cela s'exécute à nouveau — sur l'export du trimestre suivant, sur le portable de quelqu'un d'autre, sans rien modifier.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/data-analysis-foundations/foundations-07-indicator-tables