CASSION · FORMATION

Données de routine et DHIS2

Travailler avec un système d'information sanitaire agrégé — ses éléments de données, sa hiérarchie d'unités d'organisation et sa logique de périodes — et répondre à la question qui suit toujours : qu'a-t-on exactement compté ?

Formateur	Pierre Robentz Cassion
Niveau	Intermédiaire
Heures apprenant	14 h
Enseignée en	Python · R
Mise à jour	28 juillet 2026
Secteurs	Santé publique · Nutrition
Normes et méthodologies	Définitions d'indicateurs de l'UNICEF · Objectifs de développement durable (ODD) · Gestion axée sur les résultats (GAR)

Consulter et exécuter la formation en ligne sur data-analysis.cassion.dev/fr/cours/routine-data-and-dhis2

Ce que vous saurez faire

— Lire une extraction DHIS2 comme la base de données dont elle vient — éléments de données, combinaisons de catégories, unités d'organisation et périodes — plutôt que comme un tableur inexpliqué

— Agréger le long d'une hiérarchie d'unités d'organisation sans compter deux fois une formation sanitaire qui a changé de district

— Traiter la complétude et la promptitude du rapportage comme des dénominateurs, et dire ce que signifie un taux de couverture quand un quart des unités sont muettes

— Extraire des données via l'API Web avec un script qui tourne chaque mois sans retouche et consigne ce qu'il a demandé

— Rapprocher un agrégat de routine d'une estimation d'enquête portant sur la même chose, et décomposer l'écart entre mesure et sélection

— Répondre à ce qui a été compté, par qui, sur quelle période, pour tout chiffre que le système produit

Prérequis

Aucun. Cette formation ne suppose aucune expérience préalable de la programmation.

Sommaire

À propos de cette formation

La plupart des chiffres que rapporte ce secteur sortent d'un système d'information sanitaire agrégé, et dans la plupart des pays ce système est DHIS2. Chaque cours antérieur de ce programme y a emprunté — l'extraction vaccinale a servi dans cinq d'entre eux — sans jamais expliquer d'où elle venait.

Ce cours l'explique. Non comme une formation logicielle — vous ne cliquerez pas dans une interface. Comme **le modèle de données sous l'export**, car savoir qu'une valeur est un élément de données croisé avec une combinaison de catégories, rattaché à une unité d'organisation, dans une période, issu d'un ensemble de données, est ce qui transforme un tableur inexpliqué en quelque chose de défendable.

La question autour de laquelle le cours est bâti est celle qui suit tout chiffre de routine jamais présenté : **qu'a-t-on exactement compté ?** Y répondre exige quatre choses que l'export ne porte généralement pas — quel élément de données, quelles unités d'organisation, quelle logique de périodes, et combien des unités censées rapporter l'ont fait.

Les exemples s'exécutent sur l'extraction vaccinale de type DHIS2 — 38 formations sanitaires, douze mois, six antigènes, et 642 de ses 2 736 couples formation-mois ne portant aucun rapport. Et le cours se termine sur une comparaison qui donne trois réponses réelles à une seule question : la malnutrition aiguë mesurée à 8,7 % par le registre de dépistage de routine, à 11,4 % par une enquête ménage employant la même mesure, et à 14,9 % par une enquête SMART en employant une autre.

Python et R tout du long, avec les exemples d'API en Python, car c'est là que vivent la plupart des scripts d'extraction de ce secteur.

Vous devriez avoir suivi *Évaluation de la qualité des données*, qui enseigne déjà la complétude et la promptitude du rapportage comme indicateurs. Ce cours ne les répète pas ; il montre comment le système les calcule, et ce qu'il ne vous dit pas.

Première partie

Comment la base est structurée

CHAPITRE 1

Ce qu'est réellement une valeur dans DHIS2

Leçon 1 sur 8 · 90 min

Cinq coordonnées, pas un nombre. Élément de données, combinaison d'options de catégories, unité d'organisation, période et ensemble de données — et la différence entre un élément de données et un indicateur, d'où part l'essentiel de la confusion sur les chiffres de routine.

1.1 Une valeur a cinq coordonnées

L'extraction vaccinale ressemble à un tableur — formation sanitaire, période, antigène, doses. Dans le système d'origine, une valeur stockée est adressée par cinq choses, et les connaître toutes les cinq est ce qui rend un chiffre défendable.

Coordonnée	Dans l'extraction	Ce à quoi elle répond
Élément de données	`antigen` plus la mesure	Ce qui a été compté
Combinaison d'options de catégories	absente	Selon quelle ventilation
Unité d'organisation	`facility_id`	Où
Période	`period`	Quand
Ensemble de données	absent	Sur quel formulaire il a été collecté

Deux des cinq manquent à l'export, et c'est le cas normal. L'essentiel de cette leçon porte sur ce que ces deux-là portent et sur les disputes que leur absence provoque.

1.2 Élément de données — la chose qui est comptée

Un **élément de données** est la mesure brute que collecte un formulaire — « doses de penta3 administrées », « doses de BCG administrées ». C'est un décompte de quelque chose, saisi par une personne, et il est stocké exactement tel que saisi.

```python
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
print(sorted(vax["antigen"].unique()))
print(vax.groupby("antigen")["doses_administered"].sum())
```

```r
library(dplyr)
library(readr)

vax <- read_csv("vaccination-coverage-2024.v1.csv")

vax |> summarise(doses = sum(doses_administered), .by = antigen)
```

Six antigènes, six éléments de données. La colonne `target_population` est une septième valeur d'une autre nature, et dans une instance réelle ce ne serait généralement pas un élément de données du tout — c'est un dénominateur, stocké soit comme un élément de données saisi une fois par an, soit comme un attribut de l'unité d'organisation.

Demandez lequel des deux. Un dénominateur stocké comme élément annuel peut être révisé rétrospectivement ; stocké comme attribut, il change pour toutes les périodes d'un coup. Cette différence explique qu'un taux de couverture calculé en janvier puis en juin puisse différer sans aucune dose nouvelle.

1.3 Combinaisons de catégories — la ventilation intégrée à l'élément

Une **combinaison de catégories** découpe un élément de données en cellules que le formulaire collecte réellement. Le penta3 pourrait être ventilé par âge (moins d'un an / un an et plus) et par sexe, donnant quatre **combinaisons d'options de catégories** par élément.

```
Data element:  Penta3 doses administered
Category combo: Age (<1, 1+) x Sex (F, M)
Stored values:  4 per org unit per period
```

C'est là que loge le contresens le plus fréquent sur un export DHIS2. Si votre extraction a une ligne par antigène, formation et mois, l'une de deux choses est vraie, et elles ne s'équivalent pas.

— **L'élément n'a aucune ventilation**, donc il y a une valeur et rien n'a été perdu.

— **L'export a agrégé les combinaisons d'options**, auquel cas la ventilation existe dans le système et votre fichier ne peut pas la retrouver.

```python
expected = (vax["facility_id"].nunique() * vax["period"].nunique()
            * vax["antigen"].nunique())
print(f"{expected} expected rows, {len(vax)} in the file")
```

```r
c(expected = n_distinct(vax$facility_id) * n_distinct(vax$period) *
             n_distinct(vax$antigen),
  actual = nrow(vax))
```

$38 \times 12 \times 6 = 2\,736$, et le fichier compte 2 736 lignes. **La grille est exactement pleine**, ce qui vous dit que l'export est au niveau de l'élément de données sans ventilation survivante. Si un rapport demande ensuite la couverture par sexe, la réponse n'est pas dans ce fichier, et elle est peut-être ou non dans le système.

1.4 Élément de données contre indicateur

La distinction qui cause le plus d'ennuis, et il vaut la peine d'être précis.

— Un **élément de données** est stocké. Quelqu'un l'a saisi.

— Un **indicateur** est calculé. Il a un numérateur, un dénominateur et un facteur, et DHIS2 l'évalue au niveau que vous demandez.

```
Indicator:   Penta3 coverage
Numerator:   #{Penta3 doses administered}
Denominator: #{Surviving infants}
Factor:      100
```

Trois conséquences qui reviennent sans cesse.

Un indicateur n'est pas stocké, donc il ne peut pas être faux dans la base. Si un taux de couverture est faux, c'est soit un élément de données, soit la définition de l'indicateur. Les deux ont des responsables et des remèdes différents.

L'indicateur est évalué au niveau demandé, et l'arithmétique est sommer-puis-diviser, non diviser-puis-moyenner. Demander la couverture au niveau du district somme numérateurs et dénominateurs sur les formations ; cela ne moyenne pas les couvertures par formation. C'est le comportement correct et c'est l'inverse de ce que fait d'ordinaire un utilisateur de tableur.

```python
district = vax[vax["report_submitted"] == True]
correct = (district["doses_administered"].sum()
           / district["target_population"].sum())
wrong = (district["doses_administered"] / district["target_population"]).mean()
print(f"ratio of sums {correct:.3f}, mean of ratios {wrong:.3f}")
```

```r
vax |>
  filter(report_submitted) |>
  summarise(ratio_of_sums = sum(doses_administered) / sum(target_population),
            mean_of_ratios = mean(doses_administered / target_population))
```

Un indicateur peut être demandé pour une période que les données ne soutiennent pas. Demandez la couverture annuelle et DHIS2 sommera douze numérateurs mensuels sur un dénominateur qui peut être une valeur annuelle ou une somme de valeurs mensuelles, selon la configuration de l'élément. Obtenir un nombre plausible ne prouve pas que l'agrégation était juste.

1.5 Ensembles de données — le formulaire, et ce que « attendu » veut dire

Un **ensemble de données** est le formulaire sur lequel un groupe d'éléments est collecté, affecté à un ensemble d'unités d'organisation, avec un type de période. C'est lui qui décide **quelles unités sont censées rapporter**, fondement de tout chiffre de complétude que le système produit.

```python
reported = vax["report_submitted"] == True
print(f"{reported.sum()} of {len(vax)} facility-month-antigen values reported")
print(f"reporting rate {reported.mean():.1%}")
```

```r
mean(vax$report_submitted)
```

76,5 %. Ce nombre n'a de sens que parce qu'une affectation d'ensemble de données dit que les 38 formations étaient censées transmettre chaque mois. **Sans l'ensemble de données, il n'y a ni dénominateur ni complétude.**

Remarquez aussi ce que l'extraction ne porte pas : le rapportage est ici tout ou rien par couple formation-mois pour les six antigènes, ce à quoi ressemble un ensemble de données unique contenant les six. Un système répartissant les antigènes sur deux ensembles montrerait des mois partiels, et cette structure serait visible dans les données.

```python
partial = (
    vax[reported].groupby(["facility_id", "period"])["antigen"].nunique()
)
print(f"{(partial < 6).sum()} facility-months with only some antigens reported")
```

```
vax |>
  filter(report_submitted) |>
  summarise(antigens = n_distinct(antigen), .by = c(facility_id, period)) |>
  filter(antigens < 6) |>
  nrow()
```

Zéro. Un ensemble de données, six éléments, transmis ensemble.

1.6 Demandez les métadonnées, pas seulement les données

La conclusion pratique de cette leçon. Quand vous demandez une extraction, demandez les métadonnées avec.

— Les **définitions des éléments de données**, y compris leur mode d'agrégation entre périodes (somme, moyenne, ou dernière valeur — cela compte énormément et personne ne le demande).

— La **combinaison de catégories** de chaque élément, pour savoir ce qui a été réduit.

— L'**affectation des ensembles de données**, pour savoir combien d'unités étaient attendues.

— Les **définitions d'indicateurs** de tout ce qui est calculé, avec les expressions de numérateur et de dénominateur.

Quatre petits fichiers, et ils transforment un export en quelque chose à partir de quoi une fiche de référence peut s'écrire. Le cours *Conception d'indicateurs* exigeait une source détaillée jusqu'au champ ; voici à quoi cela ressemble quand la source est DHIS2.

1.7 La suite

Vous savez ce qu'est une valeur et où elle se situe. La leçon suivante la déplace — vers le haut d'une hiérarchie d'unités d'organisation qui n'est pas fixe, où une formation réaffectée d'un district à l'autre en mars donne un total de district qu'on ne peut pas simplement sommer.

CHAPITRE 2

La hiérarchie qui bouge sous vos pieds

Leçon 2 sur 8 · 90 min

Unités d'organisation, niveaux et groupes, et la propriété dont personne ne vous prévient — l'arbre est courant et non historique, si bien qu'une formation réaffectée en mars réécrit en silence les totaux de district de l'an dernier.

2.1 Un arbre, et tout y est accroché

Toute valeur du système est rattachée à une **unité d'organisation**, et les unités forment un arbre unique — pays, région, district, formation sanitaire. Le niveau où siège une unité est une propriété de l'arbre, non une colonne de l'unité.

```
Level 1  Country
Level 2  Region
Level 3  District
Level 4  Facility        <- our 38 units
```

L'agrégation est le parcours de l'arbre vers le haut. Demandez un chiffre de district et le système somme toutes les formations situées dessous ; demandez un chiffre national et il somme tous les districts. C'est la même discipline « agréger à la granularité » qu'enseignait le cours sur les jointures, avec une granularité fournie par une hiérarchie que quelqu'un d'autre entretient.

```python
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
print(f"{vax['facility_id'].nunique()} facilities in the extract")
```

```r
library(dplyr)
n_distinct(vax$facility_id)
```

Trente-huit, toutes au niveau 4. **L'extraction ne porte aucun parent**, ce qui est le cas habituel et la raison pour laquelle la plupart des totaux de district de ce secteur se calculent en joignant une liste de formations plutôt que par le système.

2.2 Les groupes ne sont pas des niveaux

L'extraction a une colonne `facility_type`, et il vaut la peine d'être clair sur ce qu'elle est.

```python
by_type = (
    vax.groupby("facility_type")["facility_id"].nunique().sort_values(ascending=False)
)
print(by_type)
```

```r
vax |> summarise(facilities = n_distinct(facility_id), .by = facility_type)
```

Type de formation	Formations
Poste de santé	20
Centre de santé	16
Hôpital de district	2

C'est un **groupe d'unités d'organisation**, non un niveau. La distinction compte de trois façons pratiques.

— **Une unité a exactement un parent et un nombre quelconque de groupes.** Poste de santé est un groupe ; cela ne dit pas où se trouve la formation.

— **Les groupes peuvent se chevaucher.** Une formation peut être à la fois « poste de santé », « difficile d'accès » et « appuyée par le partenaire X », et agréger sur deux groupes qui se chevauchent compte deux fois.

— **Les jeux de groupes sont la version sûre.** Un jeu de groupes est un ensemble de groupes mutuellement exclusifs — le type de formation en est généralement un — et agréger sur un jeu de groupes est sûr là où agréger sur un groupe quelconque ne l'est pas.

Demandez si ce sur quoi vous regroupez est un jeu de groupes. Si ce n'en est pas un, vérifiez que les groupes partitionnent avant de sommer.

2.3 La propriété dont personne ne vous prévient

Voici celle qui coûte un trimestre.

DHIS2 stocke la hiérarchie courante, non une hiérarchie historique. Une valeur est rattachée à une formation. La formation est rattachée à un parent *maintenant*. Il n'existe, dans le modèle de données ordinaire, aucune trace du parent qu'elle avait en mars.

Aussi, lorsqu'une formation est réaffectée du district A au district B — révision de limites, réorganisation, correction — **toutes ses données historiques la suivent.** Le total du district A pour janvier dernier, déjà rapporté et déjà dans le tableur d'un bailleur, change dès que quelqu'un le redemande.

Trois symptômes, tous d'allure de problèmes de qualité et aucun n'en étant un.

— Un total de district différent de la même requête lancée six mois plus tôt, sans aucune saisie entre-temps.

— Un chiffre annuel publié qu'on ne peut pas reproduire.

— Deux rapports de la même période qui divergent, tous deux exacts le jour où ils ont été produits.

2.4 Que faire

Vous ne pouvez pas empêcher la hiérarchie de bouger. Vous pouvez l'empêcher de déplacer vos chiffres en silence.

Figez la hiérarchie avec l'extraction. Tirez la liste des unités d'organisation en même temps que les données et stockez-la à côté des valeurs.

```
org_units = pd.read_csv("outputs/extract/2026-07/org-units.csv")
    # ea/facility id, name, parent_id, level, extracted_on

pinned = vax.merge(org_units[["facility_id", "district_id"]],
                   on="facility_id", how="left", validate="many_to_one")
assert pinned["district_id"].notna().all(), "facility with no district in the pinned tree"
```

```r
org_units <- readr::read_csv(here::here("outputs", "extract", "2026-07", "org-units.csv"))

pinned <- vax |>
  left_join(select(org_units, facility_id, district_id), by = "facility_id",
            relationship = "many-to-one")

stopifnot(!any(is.na(pinned$district_id)))
```

La jointure est `many_to_one` et l'assertion n'est pas décorative. Une formation présente dans les données et absente de l'arbre figé signifie que l'arbre a été tiré à un autre moment que les données, précisément la situation que ceci doit prévenir.

Agrégez depuis l'arbre figé, pas depuis l'arbre vivant. Un chiffre publié en mars est alors reproductible en septembre, car l'arbre qu'il a employé est sur disque.

Consignez la date d'extraction sur les deux. L'avant-dernière leçon en fait une propriété du script d'extraction plutôt qu'une discipline.

2.5 La réaffectation, reconstituée

Là où la date de réaffectation est connue — et quelqu'un la connaît toujours, même si le système ne la connaît pas — le traitement honnête consiste à la porter explicitement.

```python
reassignments = pd.DataFrame([
    {"facility_id": "FAC017", "from_district": "D01", "to_district": "D02",
     "effective": "2024-04-01", "reason": "boundary revision"},
])

def district_at(facility, period, log=reassignments):
    moves = log[(log["facility_id"] == facility)
                & (pd.to_datetime(log["effective"]) <= period)]
    return moves["to_district"].iloc[-1] if len(moves) else None
```

```r
reassignments <- tibble::tribble(
  ~facility_id, ~from_district, ~to_district, ~effective,     ~reason,
  "FAC017",     "D01",          "D02",        as.Date("2024-04-01"), "boundary revision"
)
```

C'est une dimension à évolution lente, et la traiter comme telle est la bonne réponse. C'est aussi plus de travail que la plupart des équipes n'en feront, si bien que le repli est un étiquetage honnête.

> Les totaux de district sont calculés sur la hiérarchie d'unités d'organisation au 28/07/2026. Une formation a été réaffectée de D01 à D02 en avril 2024 ; ses données 2024 figurent sous D02 sur toute l'année, y compris pour les périodes antérieures à la réaffectation.

Cette phrase est le livrable. Elle ne coûte rien et transforme un chiffre irreproductible en chiffre reproductible.

2.6 Agréger vers le haut sans compter deux fois

Deux modes de défaillance quand vous faites l'agrégation vous-même.

Une formation apparaissant deux fois dans l'arbre. Impossible dans DHIS2 — une unité a un parent — mais tout à fait possible dans le CSV de formations que quelqu'un vous

a envoyé, où une formation ayant déménagé figure sous les deux districts.

```python
duplicated = org_units["facility_id"].duplicated(keep=False)
assert not duplicated.any(), org_units.loc[duplicated, ["facility_id", "district_id"]]
```

```r
stopifnot(!any(duplicated(org_units$facility_id)))
```

Sommer entre niveaux. Un total de district plus ses formations, c'est le district compté deux fois. Cela arrive quand une extraction mélange les niveaux — demandez le seul niveau 4, et vérifiez.

```python
print(org_units["level"].value_counts())
```

```r
org_units |> count(level)
```

Toute extraction devrait être à exactement un niveau. Si ce n'est pas le cas, la première chose que fait votre script est de filtrer sur un seul, en disant lequel.

2.7 La couverture est un indicateur de niveau district

Un point du cours sur les enquêtes qui arrive ici avec son mécanisme. Les bassins de desserte se chevauchent et les gens traversent les limites pour se faire vacciner, si bien qu'un taux de couverture par formation est un numérateur d'une population sur un dénominateur d'une autre.

```python
by_facility = (
    vax[vax["report_submitted"] == True]
    .query("antigen == 'penta3'")
    .groupby("facility_id")
    .agg(doses=("doses_administered", "sum"), target=("target_population", "sum"))
)
by_facility["coverage"] = by_facility["doses"] / by_facility["target"]
print(by_facility["coverage"].describe()[["min", "max"]].round(3))
```

```r
vax |>
  filter(report_submitted, antigen == "penta3") |>
  summarise(coverage = sum(doses_administered) / sum(target_population),
            .by = facility_id) |>
  summarise(min = min(coverage), max = max(coverage))
```

La dispersion entre formations est bien plus large que toute différence réelle de prestation, car la traversée de limites est un bruit au niveau de la formation et s'annule au niveau du district. **Agrégez au niveau où la traversée se produit à l'intérieur de l'unité**, et rapportez les chiffres par formation comme une charge de travail plutôt que comme une couverture.

2.8 La suite

Vous savez placer une valeur dans l'espace. L'unité suivante la place dans le temps — types de période, à quelle période appartient une saisie tardive, et l'enregistrement de complétude qui décide si la valeur était attendue.

Deuxième partie

Périodes et complétude

CHAPITRE 3

Les périodes, et l'opérateur d'agrégation que personne ne règle

Leçon 3 sur 8 · 90 min

Un fichier, un indicateur, trois taux de couverture annuels — 6,5 %, 60,8 % et 77,5 % — tous issus de phrases qui sonnent défendables. La différence est la manière dont le dénominateur s'agrège dans le temps.

3.1 Types de période, et l'identifiant qui les porte

Toute valeur appartient à une période, et le type de période est une propriété de l'ensemble de données sur lequel elle a été collectée. Les identifiants de période de DHIS2 encodent le type dans la chaîne, ce qu'il vaut la peine d'adopter même hors DHIS2.

```
202408      monthly
2024W32     weekly
2024Q3      quarterly
2024        yearly
2024April   financial year starting April
```

Ils se trient correctement, ils ne peuvent pas être ambigus entre conventions, et le type se lit sans schéma. Le cours sur les jointures plaidait pour des clés de période ISO ; c'est le même argument avec le vocabulaire du système.

```python
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
vax["period_id"] = vax["period"].dt.strftime("%Y%m")
print(sorted(vax["period_id"].unique())[:4])
```

```r
library(dplyr)
vax <- vax |> mutate(period_id = format(period, "%Y%m"))
```

3.2 À quelle période appartient une valeur ?

La question a une bonne réponse et deux mauvaises d'usage quotidien.

— **Juste — la période où l'activité a eu lieu.** Une dose administrée le 28 août est d'août, quelle que soit la date d'arrivée du formulaire.

— **Faux — la période où le formulaire a été transmis.** Fréquent là où la saisie se fait par lots, et cela décale l'activité d'un mois partout où le rapportage est tardif.

— **Faux — la période où la valeur a été saisie.** Même défaillance, un cran plus loin.

DHIS2 stocke la valeur contre la période du formulaire de saisie ouvert, si bien que le système a raison *si la personne a ouvert le bon formulaire*. La défaillance est humaine et se

manifeste par un motif caractéristique — un mois bas suivi d'un mois haut, le pic-et-creux que le cours d'évaluation vous a appris à chercher.

3.3 Les trois chiffres annuels

Voici la leçon. Prenez le penta3 sur l'année et demandez la couverture annuelle.

```python
penta3 = vax[vax["antigen"] == "penta3"]
reported = penta3[penta3["report_submitted"] == True]

doses = reported["doses_administered"].sum()

summed_denominator = reported["target_population"].sum()
annual_target = penta3.groupby("facility_id")["target_population"].first().sum()

print(f"doses: {doses:,}")
print(f"a) doses / summed monthly target : {doses / summed_denominator:.1%}")
print(f"b) doses / annual target, all 38 : {doses / annual_target:.1%}")
```

```r
penta3 <- vax |> filter(antigen == "penta3")
reported <- penta3 |> filter(report_submitted)

doses <- sum(reported$doses_administered)
summed <- sum(reported$target_population)
annual <- penta3 |> summarise(t = first(target_population), .by = facility_id) |>
  summarise(sum(t)) |> pull()

c(a = doses / summed, b = doses / annual)
```

Phrase	Chiffre
« Doses sur population cible, sommée sur l'année »	**6,5 %**
« Doses annuelles sur population cible annuelle »	**60,8 %**
« Doses annuelles sur cible annuelle, parmi les couples formation-mois ayant rapporté »	**77,5 %**

Trois nombres, un fichier, un indicateur, et chacun sort d'une phrase que quelqu'un prononcerait en réunion sans sourciller.

3.4 Pourquoi ils diffèrent

6,5 % est un chiffre mensuel portant une étiquette annuelle. `target_population` est une cohorte *annuelle* — les nourrissons survivants du bassin pour l'année — et l'extraction la répète chaque mois. La sommer sur douze mois produit un dénominateur douze fois trop grand. Le résultat est la couverture mensuelle moyenne, et la multiplier par douze ramène à 78 %.

60,8 % compte les formations muettes comme n'ayant vacciné personne. Le dénominateur est la cohorte annuelle complète de chaque formation, et le numérateur ne contient que les doses des couples formation-mois ayant rapporté. 107 des 456 couples penta3 manquent, et ce chiffre leur attribue zéro dose.

77,5 % est le défendable, et il exige un ajustement explicite du dénominateur.

```python
months_reported = reported.groupby("facility_id").size()
targets = penta3.groupby("facility_id")["target_population"].first()
```

```python
prorated = (targets * months_reported.reindex(targets.index).fillna(0) / 12).sum()

print(f"c) pro-rated denominator: {doses / prorated:.1%}")
```

```r
months <- reported |> summarise(m = n(), .by = facility_id)
targets <- penta3 |> summarise(t = first(target_population), .by = facility_id)

prorated <- targets |>
  left_join(months, by = "facility_id") |>
  mutate(m = coalesce(m, 0)) |>
  summarise(sum(t * m / 12)) |> pull()

doses / prorated
```

Chaque formation contribue la part de sa cohorte annuelle correspondant aux mois qu'elle a effectivement rapportés. C'est une couverture parmi les couples formation-mois ayant rapporté, et le libellé doit le dire.

3.5 L'opérateur d'agrégation

Sous ces trois chiffres se trouve un réglage de configuration que la plupart des analystes ne voient jamais — **la façon dont un élément de données s'agrège entre périodes.**

Opérateur	Sens	Convient à
Somme	Additionner entre périodes	Décomptes d'événements — doses, admissions, consultations
Moyenne	Moyenne entre périodes	Niveaux de stock, effectifs, tout ce qui est un état et non un événer
Dernière valeur	Prendre la plus récente	Populations, cibles, tailles de registre

`doses_administered` se somme. `target_population` **ne doit pas** — c'est un état, et il se somme à douze fois lui-même. Dans une instance réelle, l'élément serait configuré en moyenne ou en dernière valeur, et demander la couverture annuelle fonctionnerait. Ici c'est une colonne répétée et la correction vous revient.

Demandez l'opérateur d'agrégation avec les métadonnées. C'est un champ par élément de données, invisible dans l'export, et c'est la différence entre 6,5 % et 78 %.

Le même réglage existe pour l'agrégation entre unités d'organisation, où « somme » est presque toujours juste et « moyenne » presque toujours fausse — les doses d'un district sont la somme de celles de ses formations, non leur moyenne.

3.6 Échéances, expiration et verrouillage

Trois réglages qui déterminent si une valeur peut encore changer.

— **L'échéance** est la date à laquelle l'ensemble de données est dû. Elle alimente le chiffre de promptitude que calcule le cours d'évaluation.

— **Les jours d'expiration** verrouillent le formulaire un nombre fixe de jours après la fin de la période. Ensuite, la saisie exige un déverrouillage.

— **Une exception de verrouillage** déverrouille un ensemble, une unité, une période. Chacune est une décision, et un système en comptant des centaines n'a en pratique aucun verrouillage.

Conséquence pour un analyste — **une extraction d'une période récente est provisoire.** Tirez août en septembre puis en novembre et les nombres différeront, légitimement, parce que la saisie tardive continue d'arriver. Ce n'est pas un problème de qualité et cela ne doit pas être rapporté comme tel — c'est la raison pour laquelle l'avant-dernière leçon consigne une date d'extraction à chaque tirage.

3.7 La complétude est un enregistrement à part

Dernier point structurel, et il explique un nombre qui paraît sinon incohérent.

La complétude dans DHIS2 est un enregistrement, non un calcul. Quand un utilisateur clique « complet » sur un formulaire de saisie, le système stocke un enregistrement de complétude pour cet ensemble, cette unité et cette période. Le taux de rapportage est construit à partir de ces enregistrements.

Ce qui signifie qu'un ensemble peut être **complet sans aucune valeur** — quelqu'un a cliqué complet sur un formulaire vide — et **plein de valeurs sans être complet** — données saisies, personne n'a cliqué. Les deux arrivent constamment.

```python
grid = (vax["facility_id"].nunique() * vax["period"].nunique()
        * vax["antigen"].nunique())
print(f"{grid} rows expected, {len(vax)} present, "
      f"{(vax['report_submitted'] == True).sum()} flagged reported")
```

```r
c(expected = n_distinct(vax$facility_id) * n_distinct(vax$period) * n_distinct(vax$antigen
    ),
  present = nrow(vax),
  reported = sum(vax$report_submitted))
```

2 736 lignes présentes et 2 094 marquées comme rapportées. **La ligne existe dans les deux cas**, exactement la forme contre laquelle le cours de nettoyage mettait en garde — un rapport manquant qui arrive sous forme de zéro avec un indicateur à côté.

3.8 La suite

L'indicateur de rapportage est la matière première du chiffre de complétude, et la leçon suivante en fait un dénominateur. Le cours d'évaluation a déjà montré pourquoi cela compte ; celui-ci montre comment le système le calcule, et les deux façons dont sa réponse diffère de la vôtre.

CHAPITRE 4

Le taux de rapportage est un dénominateur, pas une note de bas de page

Leçon 4 sur 8 · 75 min

Le système calcule quatre mesures de complétude et elles divergent. Celle que cite votre rapport décide si un taux de couverture est une estimation ou une borne inférieure, et août vaut 29 % dans les deux cas.

4.1 Quatre mesures, un seul mot

Le cours *Évaluation de la qualité des données* a établi que la complétude du rapportage est un indicateur à part entière. Cette leçon porte sur ce que le système entend par là, car DHIS2 expose quatre mesures voisines sous des noms employés indifféremment.

Mesure	Numérateur	Dénominateur
Taux de rapportage	Ensembles marqués complets	Ensembles attendus
Rapports effectifs	Ensembles marqués complets	— (un décompte)
Rapports attendus	—	Unités affectées × périodes
Taux de rapportage à temps	Complétés avant l'échéance	Ensembles attendus

Le premier et le dernier ne diffèrent que par la promptitude, et un rapport citant « le taux de rapportage » sans préciser lequel laisse le lecteur incapable de dire si 82 % veut dire « transmis » ou « transmis à temps ».

```python
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
vax["reported"] = vax["report_submitted"] == True

facility_months = vax.drop_duplicates(["facility_id", "period"])
print(f"expected: {len(facility_months)}")
print(f"actual:   {facility_months['reported'].sum()}")
print(f"rate:     {facility_months['reported'].mean():.1%}")
```

```r
library(dplyr)

vax |>
  distinct(facility_id, period, .keep_all = TRUE) |>
  summarise(expected = n(), actual = sum(report_submitted),
            rate = mean(report_submitted))
```

456 couples formation-mois attendus, 349 transmis, **76,5 %**. Notez le dédoublonnage — la complétude est enregistrée par ensemble de données, unité et période, non par élément de données, si bien que compter les 2 736 lignes au niveau élément répondrait à une autre question et donnerait le même pourcentage par coïncidence.

4.2 Là où la réponse du système diffère de la vôtre

Deux différences, et les deux ont pris des gens en défaut.

Les rapports attendus comptent l'affectation, non l'existence. Une formation fermée en mars reste au dénominateur le reste de l'année tant que personne ne désaffecte l'ensemble de données ou ne ferme l'unité. Les taux de rapportage dérivent donc vers le bas à mesure que des formations ferment sans mise à jour des affectations, et cette dérive ressemble à une performance qui se dégrade.

La complétude est un clic, non un calcul. Une formation peut saisir toutes ses valeurs sans jamais marquer le formulaire complet ; le système la compte comme non déclarante alors que les données sont là. L'inverse arrive aussi. Le taux de rapportage et la présence de données sont donc deux questions différentes.

```python
has_data = (
    vax.groupby(["facility_id", "period"])["doses_administered"]
    .sum().gt(0).rename("has_data")
)
flags = facility_months.set_index(["facility_id", "period"])["reported"]

crosstab = pd.crosstab(flags, has_data.reindex(flags.index))
print(crosstab)
```

```r
vax |>
  summarise(reported = first(report_submitted),
            has_data = sum(doses_administered) > 0,
            .by = c(facility_id, period)) |>
  count(reported, has_data)
```

Sur cette extraction les deux concordent exactement — tout mois rapporté a des doses et tout mois non rapporté n'en a aucune — ce qui est le cas net. **Faites le tableau croisé quand même.** Les cellules hors diagonale sont là où une instance réelle garde ses problèmes les plus intéressants, et une cellule « marqué complet, aucune donnée » est une formation qui transmet des formulaires vides pour atteindre une cible.

4.3 Le taux de rapportage comme dénominateur de tout le reste

Le point opérationnel, et la raison pour laquelle cette leçon figure dans un cours DHIS2 et pas seulement dans celui d'évaluation.

```python
monthly = (
    vax[vax["antigen"] == "penta3"]
    .groupby(vax["period"].dt.strftime("%Y-%m"))
    .apply(lambda g: pd.Series({
        "reporting_rate": g["reported"].mean(),
        "coverage_reporting": (g.loc[g["reported"], "doses_administered"].sum()
                               / g.loc[g["reported"], "target_population"].sum()),
    }))
)
print((monthly * 100).round(1))
```

```r
vax |>
  filter(antigen == "penta3") |>
  mutate(month = format(period, "%Y-%m")) |>
  summarise(
```

```
    reporting_rate = mean(report_submitted),
    coverage = sum(doses_administered[report_submitted]) /
               sum(target_population[report_submitted]),
    .by = month
  )
```

Mois	Taux de rapportage	Couverture parmi les déclarants
Juillet	82 %	…
Août	**29 %**	…
Septembre	**45 %**	…
Octobre	95 %	…

Deux colonnes, toujours adjacentes. Le taux de couverture d'août est calculé sur onze des trente-huit formations, et quoi qu'il dise, il le dit de ces onze-là.

Trois règles en découlent, et elles sont le livrable de cette leçon.

— **En dessous de 90 % de rapportage, un taux de couverture est une borne inférieure**, non une estimation. Étiquetez-le « parmi les formations ayant rapporté », à chaque fois.

— **Ne comparez pas deux périodes aux taux de rapportage sensiblement différents** sans le dire. Août face à juillet compare onze formations à trente et une.

— **Une courbe de tendance exige le taux de rapportage sur le même graphique.** Tracée seule, une série de couverture traversant août ressemble à un effondrement de programme ; tracée avec la complétude, l'explication est immédiate.

4.4 Les deux façons de traiter une formation muette

Aucune n'est fausse. Elles répondent à des questions différentes et doivent être étiquetées.

L'exclure du numérateur et du dénominateur. Couverture parmi les formations ayant rapporté. Honnête, et c'est ce que fait le code ci-dessus.

Imputer ses doses. Remplir à partir des mois récents de la formation, en le disant. C'est ce que font les estimations nationales — WUENIC et exercices similaires imputent plutôt que de laisser des trous, car la question porte sur des enfants et non sur de la paperasse.

```
by_facility = (
    vax[(vax["antigen"] == "penta3") & vax["reported"]]
    .groupby("facility_id")["doses_administered"].median()
)
missing = (vax["antigen"] == "penta3") & ~vax["reported"]
imputed_total = (vax.loc[missing, "facility_id"].map(by_facility).sum()
                 + vax.loc[(vax["antigen"] == "penta3") & vax["reported"],
                           "doses_administered"].sum())

annual_target = (vax[vax["antigen"] == "penta3"]
                 .groupby("facility_id")["target_population"].first().sum())
print(f"imputed annual coverage: {imputed_total / annual_target:.1%}")
```

```
median_by_facility <- vax |>
  filter(antigen == "penta3", report_submitted) |>
  summarise(m = median(doses_administered), .by = facility_id)
```

N'imputez jamais en silence. Un chiffre imputé et un chiffre rapporté dans la même colonne sans marquage sont le défaut que le cours de nettoyage a nommé — une estimation qu'on ne distingue plus d'une mesure.

4.5 Rapportez la paire

```python
summary = pd.DataFrame({
    "period": ["2024-08", "2024-09", "2024-10"],
    "facilities_reporting": [11, 17, 36],
    "facilities_expected": [38, 38, 38],
    "reporting_rate": ["29%", "45%", "95%"],
    "coverage_basis": ["among reporting facilities"] * 3,
})
```

```r
tibble::tribble(
  ~period,   ~reporting, ~expected, ~basis,
  "2024-08",         11,        38, "among reporting facilities",
  "2024-09",         17,        38, "among reporting facilities",
  "2024-10",         36,        38, "among reporting facilities"
)
```

Cinq colonnes, et aucun taux de couverture ne circule sans elles. C'est tout l'argument, et c'est moins cher que la réunion où quelqu'un demande pourquoi août s'est effondré.

4.6 La suite

Tout ce qui précède a tourné sur un CSV exporté à la main. L'unité suivante le remplace — l'API Web, ce qu'il faut lui demander, et un script qui fait la même requête chaque mois en consignant ce qu'il a demandé.

Troisième partie

Sortir les données

CHAPITRE 5

Interroger le système au lieu d'exporter à la main

Leçon 5 sur 8 · 90 min

Deux points d'accès qui répondent à des questions différentes, la syntaxe de dimensions qui adresse un cube, et les quatre choses à régler avant tout — authentification, pagination, limites de débit, et le fait que les tables analytiques sont périmées tant que personne ne les reconstruit.

5.1 L'export mensuel est le problème

Quelqu'un ouvre l'interface, choisit les unités d'organisation, les périodes, les éléments de données, clique sur exporter, et envoie un tableur par courriel. Le mois suivant, quelqu'un recommence, légèrement différemment, et personne ne sait dire comment.

C'est ce processus que cette leçon remplace. Non parce que l'interface est mauvaise — c'est ainsi que la plupart des gens devraient employer DHIS2 — mais parce qu'**une analyse qui commence par un fichier bâti à la main n'est pas reproductible**, et la reproduire était tout l'objet du module 2.

5.2 Deux points d'accès, deux questions

La chose la plus utile à savoir sur l'API est qu'il existe deux façons de sortir des données et qu'elles répondent à des questions différentes.

Point d'accès	Renvoie	À employer pour
`/api/dataValueSets`	Les valeurs brutes stockées, telles que saisies	Reproduire le registre ; tout ce qui exige de v
`/api/analytics`	Des valeurs agrégées et calculées	Indicateurs, totaux remontés dans la hiérarch

La différence compte plus qu'il n'y paraît.

`dataValueSets` donne la vérité telle que stockée — une ligne par élément de données, unité d'organisation, période et combinaison d'options de catégories. Aucune agrégation, aucune arithmétique d'indicateur, aucun remplissage. C'est ce qu'il vous faut quand l'analyse doit être défendable jusqu'à la saisie.

`analytics` donne la réponse que le système afficherait sur un tableau de bord — agrégée dans la hiérarchie, indicateurs évalués, opérateurs d'agrégation appliqués. C'est ce qu'il vous faut quand il vous faut le même chiffre que celui que regarde le ministère.

Tirez les deux quand ils sont censés concorder. Là où ils divergent, l'écart est une règle d'agrégation dont vous n'avez pas tenu compte, et la trouver vaut une après-midi.

5.3 Adresser le cube

Les requêtes `analytics` sont dimensionnelles. Vous demandez une tranche de cube, et chaque dimension porte un nom.

```
GET /api/analytics.json
  ?dimension=dx:PENTA3_UID;BCG_UID
```

```
  &dimension=ou:LEVEL-4;DISTRICT_UID
  &dimension=pe:202401;202402;202403
  &displayProperty=NAME
```

— `dx` — dimension de données : éléments de données, indicateurs, opérandes.

— `ou` — unités d'organisation. `LEVEL-4` désigne toutes les unités du niveau 4 ; y adjoindre l'UID d'un district restreint à ce sous-arbre.

— `pe` — périodes, explicites (`202401`) ou relatives (`LAST_12_MONTHS`).

Préférez les périodes explicites aux relatives dans un script. `LAST_12_MONTHS` renvoie une fenêtre différente chaque mois, ce qui est commode pour un tableau de bord et fatal pour la reproductibilité. Calculez la fenêtre dans votre code, puis demandez-la nommément, pour que la requête elle-même consigne ce qui a été demandé.

```python
import requests

BASE = "https://play.dhis2.org/40/api"

def analytics(dx, ou, pe, session):
    response = session.get(
        f"{BASE}/analytics.json",
        params={
            "dimension": [f"dx:{';'.join(dx)}", f"ou:{ou}", f"pe:{';'.join(pe)}"],
            "displayProperty": "NAME",
            "skipMeta": "false",
        },
        timeout=120,
    )
    response.raise_for_status()
    return response.json()
```

```r
library(httr2)

analytics <- function(dx, ou, pe) {
  request("https://play.dhis2.org/40/api/analytics.json") |>
    req_url_query(
      dimension = paste0("dx:", paste(dx, collapse = ";")),
      dimension = paste0("ou:", ou),
      dimension = paste0("pe:", paste(pe, collapse = ";")),
      displayProperty = "NAME",
      .multi = "explode"
    ) |>
    req_perform() |>
    resp_body_json()
}
```

`skipMeta=false` est délibéré. Le bloc de métadonnées porte les noms derrière chaque UID de la réponse, et sans lui vous avez un tableau d'identifiants que personne ne peut lire.

5.4 La forme de la réponse

`analytics` renvoie un bloc d'en-têtes, un tableau de lignes et un dictionnaire de métadonnées. Transformez-le en tableau immédiatement et joignez les noms.

```python
def to_frame(payload):
```

```
    import pandas as pd

    columns = [h["name"] for h in payload["headers"]]
    frame = pd.DataFrame(payload["rows"], columns=columns)

    names = {uid: item["name"]
             for uid, item in payload["metaData"]["items"].items()}
    for column in ("dx", "ou", "pe"):
        if column in frame:
            frame[f"{column}_name"] = frame[column].map(names)
    return frame
```

```
to_frame <- function(payload) {
  cols <- vapply(payload$headers, \(h) h$name, character(1))
  rows <- do.call(rbind, lapply(payload$rows, \(r) unlist(r)))
  as.data.frame(rows, stringsAsFactors = FALSE) |> setNames(cols)
}
```

Conservez les UID en plus des noms. Les noms changent ; les UID non. Un script qui joint sur un nom casse à la première correction orthographique, et il casse en silence, en renvoyant moins de lignes.

5.5 Les quatre choses à régler d'abord

L'authentification. Employez un jeton d'accès personnel là où l'instance le permet, l'authentification de base sinon, et **ne mettez jamais l'un ni l'autre dans le script.** Une variable d'environnement ou un fichier d'identifiants hors du dépôt, toujours.

```
import os

session = requests.Session()
session.headers["Authorization"] = f"ApiToken {os.environ['DHIS2_TOKEN']}"
```

```
req_headers(request(url), Authorization = paste("ApiToken", Sys.getenv("DHIS2_TOKEN")))
```

Un dépôt d'analyse contenant un identifiant valide pour un système d'information sanitaire national est un incident grave, et cela s'est produit assez souvent pour que ce soit la première chose qu'un relecteur doive chercher.

La pagination. Les points d'accès de métadonnées paginent par défaut à 50. Les points d'accès de données renverront volontiers des millions de lignes et expireront avant.

```
def paged(url, session, page_size=1000):
    page = 1
    while True:
        payload = session.get(url, params={"page": page, "pageSize": page_size},
                              timeout=120).json()
        yield payload
        pager = payload.get("pager", {})
        if page >= pager.get("pageCount", 1):
            return
        page += 1
```

```
# Same idea: loop until page == pageCount, and never assume one request is all of it.
```

Le débit et la taille. Demandez un district et une année à la fois, non le pays et cinq années. Une requête qui répond en huit secondes vaut mieux qu'une qui expire à quatre-vingt-dix, et la boucle fait trois lignes. Soyez un bon citoyen d'un serveur où travaillent aussi des agents de saisie.

Les tables analytiques sont périmées. C'est celle qui surprend. `analytics` lit des tables pré-agrégées reconstruites selon un calendrier, en général nocturne. **Une donnée saisie aujourd'hui n'est pas dans `analytics` tant que les tables n'ont pas tourné.** `dataValueSets` la voit immédiatement.

Les deux points d'accès peuvent donc légitimement diverger, et la bonne réaction n'est pas d'ouvrir un ticket. Vérifiez la dernière exécution.

```python
info = session.get(f"{BASE}/system/info", timeout=60).json()
print(info.get("lastAnalyticsTableSuccess"))
```

```r
resp <- request(paste0(base, "/system/info")) |> req_perform() |> resp_body_json()
resp$lastAnalyticsTableSuccess
```

Consignez cet horodatage avec chaque extraction. C'est la différence entre « les chiffres ont changé » et « les chiffres ont changé parce que les tables analytiques n'avaient pas tourné quand j'ai tiré ».

5.6 Tirez aussi les métadonnées

Tout ce que la leçon 1 réclamait est un point d'accès.

```
/api/dataElements.json?fields=id,name,aggregationType,categoryCombo[id,name]
/api/organisationUnits.json?fields=id,name,level,parent[id]&paging=false
/api/dataSets.json?fields=id,name,periodType,organisationUnits~size
/api/indicators.json?fields=id,name,numerator,denominator,indicatorType[name]
```

Quatre requêtes. Elles vous donnent l'opérateur d'agrégation, la hiérarchie figée, le dénominateur des rapports attendus et les définitions d'indicateurs — soit toutes les questions que les leçons précédentes disaient qu'il faudrait poser à un collègue.

Enregistrez-les à côté des données, et la leçon suivante rend cela automatique.

5.7 La suite

Vous savez poser une question au système. La leçon suivante rend cette interrogation reproductible — un script, une fenêtre énoncée, les métadonnées tirées en même temps, et un manifeste qui consigne exactement ce qui a été demandé et quand.

CHAPITRE 6

Une extraction que l'on peut désigner

Leçon 6 sur 8 · 75 min

Un script, un dossier daté, un manifeste qui consigne ce qui a été demandé et quand, et la réponse brute conservée à côté du tableau propre — pour qu'un chiffre publié en mars soit encore défendable en septembre.

6.1 La question à laquelle cette leçon répond

Six mois après avoir publié un taux de couverture de district, quelqu'un relance la requête et obtient autre chose. Cela arrivera, et il y a quatre raisons légitimes — saisie tardive, dénominateur révisé, formation réaffectée, et tables analytiques qui n'avaient pas tourné.

Vous ne pouvez en empêcher aucune. **Vous pouvez rendre possible de dire laquelle c'était**, et cela tient entièrement à ce que vous avez conservé.

6.2 La forme

```
extracts/
  2026-07-28/
    manifest.json          what was requested, by whom, when
    raw/
      analytics.json       the response, untouched
      dataValueSets.json
      dataElements.json
      organisationUnits.json
      dataSets.json
      indicators.json
    tidy/
      coverage_by_facility_period.csv
```

Trois règles donnent sa valeur à ce dossier.

Un dossier daté par extraction, jamais écrasé. Le disque coûte moins cher que la réunion.

La réponse brute est conservée intacte. Si votre analyse syntaxique s'avère fausse, la donnée est toujours là. Un CSV propre est un artefact dérivé, et un artefact dérivé ne se dé-dérive pas.

Le manifeste est l'essentiel. Tout le reste s'en récupère.

6.3 Le manifeste

```
import json
import datetime as dt
from pathlib import Path

def write_manifest(folder, request, system_info):
    manifest = {
        "extracted_at": dt.datetime.now(dt.timezone.utc).isoformat(),
```

```
        "extracted_by": "me-officer@example.org",
        "instance": request["base_url"],
        "dhis2_version": system_info.get("version"),
        "analytics_last_run": system_info.get("lastAnalyticsTableSuccess"),
        "request": {
            "endpoint": request["endpoint"],
            "dx": request["dx"],
            "ou": request["ou"],
            "pe": request["pe"],
        },
        "rows_returned": request["rows"],
        "script_version": "extract.py 1.3",
    }
    Path(folder, "manifest.json").write_text(json.dumps(manifest, indent=2))
    return manifest
```

```
manifest <- list(
  extracted_at        = format(Sys.time(), "%Y-%m-%dT%H:%M:%SZ", tz = "UTC"),
  instance            = base_url,
  analytics_last_run  = system_info$lastAnalyticsTableSuccess,
  request             = list(endpoint = "analytics", dx = dx, ou = ou, pe = pe),
  rows_returned       = nrow(tidy),
  script_version      = "extract.R 1.3"
)

jsonlite::write_json(manifest, file.path(folder, "manifest.json"), auto_unbox = TRUE)
```

Deux champs justifient à eux seuls leur place. **`analytics_last_run`** dit si une saisie tardive a pu être incluse. **`pe`** consigne les périodes exactement demandées, ce qui rend une fenêtre relative comme « les douze derniers mois » reconstructible au lieu d'être la description du jour où le script a tourné.

Le reste est la provenance que réclamait déjà le cours *Jointures et restructuration*, arrivant ici avec un système d'où la tirer.

6.4 Le script

```
def extract(base_url, dx, ou, pe, token, out_root="extracts"):
    session = requests.Session()
    session.headers["Authorization"] = f"ApiToken {token}"

    folder = Path(out_root, dt.date.today().isoformat())
    (folder / "raw").mkdir(parents=True, exist_ok=True)
    (folder / "tidy").mkdir(exist_ok=True)

    info = session.get(f"{base_url}/system/info", timeout=60).json()

    payload = analytics(dx, ou, pe, session)
    (folder / "raw" / "analytics.json").write_text(json.dumps(payload))

    for name, path in METADATA_ENDPOINTS.items():
        meta = session.get(f"{base_url}/{path}", timeout=120).json()
        (folder / "raw" / f"{name}.json").write_text(json.dumps(meta))

    tidy = to_frame(payload)
    tidy.to_csv(folder / "tidy" / "coverage_by_facility_period.csv", index=False)

    write_manifest(folder, {"base_url": base_url, "endpoint": "analytics",
```

```
                              "dx": dx, "ou": ou, "pe": pe, "rows": len(tidy)}, info)
    return folder
```

```
extract <- function(base_url, dx, ou, pe, token) {
  folder <- file.path("extracts", Sys.Date())
  dir.create(file.path(folder, "raw"), recursive = TRUE, showWarnings = FALSE)
  dir.create(file.path(folder, "tidy"), showWarnings = FALSE)
  # ... same shape: system info, analytics, metadata, tidy, manifest
  folder
}
```

La fenêtre est calculée, puis passée en argument. L'appelant décide quels douze mois ; le script les consigne. Ce seul choix sépare une extraction que l'on peut désigner d'une extraction que l'on peut seulement relancer.

```
def last_full_months(n, today=None):
    today = today or dt.date.today()
    first_of_this = today.replace(day=1)
    months = []
    for i in range(n, 0, -1):
        m = first_of_this - dt.timedelta(days=1)
        for _ in range(i - 1):
            m = m.replace(day=1) - dt.timedelta(days=1)
        months.append(m.strftime("%Y%m"))
    return months
```

```
last_full_months <- function(n) {
  ends <- seq(Sys.Date(), by = "-1 month", length.out = n + 1)[-1]
  rev(format(ends, "%Y%m"))
}
```

Notez `last_full_months`, non `last_months`. **Le mois en cours est toujours incomplet**, et l'inclure fait paraître toute tendance en baisse — le piège de la période partielle du cours sur les jointures, arrivant avec un système qui vous la servira volontiers.

6.5 Validez à la frontière

L'extraction est une lecture dans le système de quelqu'un d'autre, ce qui en fait exactement le raccord sur lequel le cours de nettoyage posait un contrat.

```
def check(tidy, expected_org_units, expected_periods):
    problems = []
    if tidy["ou"].nunique() != expected_org_units:
        problems.append(f"{tidy['ou'].nunique()} org units, expected {expected_org_units}"
    )
    if tidy["pe"].nunique() != expected_periods:
        problems.append(f"{tidy['pe'].nunique()} periods, expected {expected_periods}")
    if tidy["value"].isna().any():
        problems.append("null values in the response")
    return problems
```

```
check <- function(tidy, expected_ou, expected_pe) {
  c(if (n_distinct(tidy$ou) != expected_ou) "unexpected org unit count",
    if (n_distinct(tidy$pe) != expected_pe) "unexpected period count")
}
```

Le plus précieux est le décompte d'unités d'organisation. **Une extraction silencieusement plus petite est la défaillance d'API la plus fréquente en pratique** — un changement de droits, une réorganisation, une unité fermée — et elle produit un total de district discrètement plus bas, sans rien dans le fichier pour dire pourquoi.

6.6 Comparez à l'extraction précédente

Deux dossiers datés rendent une comparaison possible, et c'est là que les quatre explications se séparent.

```python
def compare(old_folder, new_folder, keys=("ou", "pe", "dx")):
    old = pd.read_csv(Path(old_folder, "tidy", "coverage_by_facility_period.csv"))
    new = pd.read_csv(Path(new_folder, "tidy", "coverage_by_facility_period.csv"))

    merged = old.merge(new, on=list(keys), how="outer",
                       suffixes=("_old", "_new"), indicator=True)
    changed = merged[merged["value_old"] != merged["value_new"]]
    return {
        "in_both": int((merged["_merge"] == "both").sum()),
        "new_only": int((merged["_merge"] == "right_only").sum()),
        "gone": int((merged["_merge"] == "left_only").sum()),
        "values_changed": len(changed),
    }
```

```r
compare <- function(old, new) {
  full_join(old, new, by = c("ou", "pe", "dx"), suffix = c("_old", "_new")) |>
    summarise(changed = sum(value_old != value_new, na.rm = TRUE),
              gone = sum(is.na(value_new)), added = sum(is.na(value_old)))
}
```

Quatre nombres, et chacun désigne une cause différente. **Des valeurs modifiées** sur les mêmes clés, c'est une saisie tardive ou une révision. **Des lignes disparues**, c'est une réaffectation ou un changement de droits. **Des lignes ajoutées**, c'est du rapportage tardif qui arrive. Et un gros écart avec `analytics_last_run` qui a bougé entre-temps, ce sont les tables analytiques et non les données.

Faites-le chaque mois et conservez la sortie. C'est la série de constats du cours d'évaluation, bâtie sur une source que vous maîtrisez désormais.

6.7 Ce que cela remplace

```
Before:  a spreadsheet in an email, monthly, method unknown
After:   extracts/2026-07-28/, one command, manifest, raw responses,
         a diff against last month, and a figure you can defend in September
```

Le script fait environ quatre-vingts lignes. C'est le morceau de code au meilleur rendement de ce cours, et celui que la plupart des équipes n'écrivent jamais parce que l'export manuel marche très bien le jour même.

6.8 La suite

Vous avez un chiffre de routine que vous pouvez désigner. La dernière unité le pose à côté d'une estimation d'enquête portant sur la même chose — trois nombres pour la malnutrition

aiguë, de 8,7 % à 14,9 % — et décompose l'écart selon les deux causes qui le produisent réellement.

Quatrième partie

Le lire face à autre chose

CHAPITRE 7

Trois réponses à une seule question

Leçon 7 sur 8 · 90 min

La malnutrition aiguë vaut 8,7 % dans le registre de dépistage de routine, 11,4 % dans une enquête ménage employant la même mesure, et 14,9 % dans une enquête SMART en employant une autre. Décomposez l'écart entre sélection et mesure, et aucune source n'est fausse.

7.1 La réunion

Le cluster nutrition dispose de trois chiffres pour la malnutrition aiguë, même population et même année. Le registre de dépistage de routine dit 8,7 %. Une enquête ménage dit 11,4 %. Une enquête SMART dit 14,9 %.

Quelqu'un demandera lequel est juste. La réponse est que les trois le sont, et le travail utile consiste à décomposer l'écart selon ses deux causes — **qui a été mesuré** et **ce qui a été mesuré** — car chacune a une implication différente pour le programme.

7.2 Les trois sources

```python
import pandas as pd

muac = pd.read_csv("muac-screening-artibonite-2024.v1.csv")
measured = muac[muac["muac_mm"].notna()]
routine = ((measured["muac_mm"] < 125) | (measured["oedema"] == True)).mean()

print(f"routine screening register: {routine:.1%} on n={len(measured):,}")
```

```r
library(dplyr)

muac |>
  filter(!is.na(muac_mm)) |>
  summarise(rate = mean(muac_mm < 125 | oedema), n = n())
```

Source	Mesure	Population	Estimation	n
Registre de dépistage de routine	PB	Enfants amenés au dépistage	**8,7 %**	4 146
Enquête ménage	PB	Échantillon de population, pondéré	**11,4 %**	648
Enquête SMART	Score z poids-taille	Échantillon de population	**14,9 %**	874

Deux choses varient entre ces lignes, et elles varient une à la fois, ce qui rend la décomposition possible.

7.3 Mesure constante, sélection variable — de 8,7 % à 11,4 %

Les lignes un et deux emploient toutes deux le PB. La seule différence est qui a été mesuré.

Le registre de routine mesure **les enfants que quelqu'un a amenés à un dépistage**. L'enquête ménage mesure **des enfants tirés de la population**, avec les pondérations construites par le cours sur les enquêtes.

Un écart de 2,7 points, et il va dans le sens où il va toujours. Le dépistage atteint les enfants dont les accompagnants peuvent atteindre un point de dépistage — plus près du site, moins contraints par d'autres travaux, dans un ménage où quelqu'un peut se déplacer. Les enfants les plus difficiles à atteindre sont systématiquement plus susceptibles d'être malnutris, et le chiffre de routine ne les contient pas.

Les données de routine mesurent la population servie, non la population. Ce n'est pas un défaut du système, c'est ce qu'est un registre de service. L'erreur est de le lire comme une prévalence.

7.4 Sélection constante, mesure variable — de 11,4 % à 14,9 %

Les lignes deux et trois sont toutes deux des échantillons de population. La différence est la mesure.

Le PB et le poids-pour-taille n'identifient pas les mêmes enfants. Ils sont corrélés, mais chacun trouve des cas que l'autre manque — le PB est plus sensible chez les enfants plus jeunes et plus petits, le poids-pour-taille chez les plus âgés et plus grands. Appliquer les seuils standards à la même population donne des prévalences différentes, et dans la plupart des contextes le poids-pour-taille donne le chiffre le plus élevé.

Un écart de 3,5 points ici, et il signifie que les deux chiffres **ne peuvent pas se lire face au même seuil**. Le seuil d'urgence de 15 % pour la malnutrition aiguë globale est défini pour une mesure précise, et lui comparer une prévalence fondée sur le PB revient à comparer deux quantités différentes. Le cours sur les indicateurs le disait à propos du nommage ; voici ce que cela coûte.

```python
comparison = pd.DataFrame({
    "source": ["Routine register", "Household survey", "SMART survey"],
    "measure": ["MUAC", "MUAC", "weight-for-height z"],
    "population": ["screened", "population sample", "population sample"],
    "estimate": [0.087, 0.114, 0.149],
})
comparison["vs_previous"] = comparison["estimate"].diff()
print(comparison)
```

```r
tibble::tribble(
  ~source,             ~measure, ~population,         ~estimate,
  "Routine register",  "MUAC",   "screened",              0.087,
  "Household survey",  "MUAC",   "population sample",     0.114,
  "SMART survey",      "WHZ",    "population sample",     0.149
) |> mutate(vs_previous = estimate - lag(estimate))
```

La sélection explique 2,7 points. La mesure en explique 3,5. À elles deux elles rendent compte de la totalité de l'écart de 6,2 points, et aucune n'est une erreur.

7.5 Ce à quoi chaque source sert réellement

Source	Bonne pour	Pas
Registre de routine	Charge de cas, charge de travail, approvisionnement, tendance dans la population servie	Prév
Enquête ménage	Prévalence de population avec intervalle, équité entre strates	Tout
Enquête SMART	Prévalence face aux seuils IPC et d'urgence, comparabilité avec d'autres enquêtes	Tout

Lisez ce tableau et la réunion se règle d'elle-même. Le responsable de programme qui demande « combien d'enfants admettrons-nous le trimestre prochain » veut le registre de routine. Le cluster qui demande « la situation a-t-elle franchi le seuil d'urgence » veut l'enquête SMART. Poser l'une des questions à l'autre source produit une réponse fausse qui sonne défendable.

7.6 Triangulation, non réconciliation

Le réflexe, quand deux sources divergent, est de les réconcilier — décider laquelle a raison et ajuster l'autre. Résistez-y. **Les sources mesurent des choses différentes et la différence est une information.**

Ce que la différence vous apprend.

— **Un écart routine-enquête qui se creuse** signifie que la couverture du dépistage baisse, ou que les plus difficiles à atteindre le deviennent davantage. C'est un constat programmatique qu'aucune source seule ne montre.

— **Un chiffre de routine qui bouge alors que l'enquête ne bouge pas** relève en général du dépistage — une nouvelle sortie de proximité, un nouveau protocole — et non de l'état nutritionnel.

— **Un chiffre d'enquête qui bouge alors que la routine ne bouge pas** signifie que le programme ne voit pas le changement. C'est le plus actionnable des trois.

```python
def gap_over_time(routine_series, survey_series):
    return pd.DataFrame({
        "routine": routine_series,
        "survey": survey_series,
        "ratio": survey_series / routine_series,
    })
```

```r
gap_over_time <- function(routine, survey) {
  tibble::tibble(routine, survey, ratio = survey / routine)
}
```

Suivez le rapport, pas la différence. Le rapport enquête sur routine est à peu près l'inverse de la couverture du dépistage, et il est assez stable pour être un indicateur de suivi à part entière.

7.7 La phrase du rapport

```
Acute malnutrition in the district, 2024:

  14.9% (95% CI 12.3-17.9) by weight-for-height z-score, from a 30-cluster
  SMART survey of 874 children, design effect 1.5.

  11.4% by MUAC in the same population, from a household survey of 648
```

```
   measured children, weighted to the sampling frame.

   8.7% by MUAC among 4,146 children brought to routine screening. This is a
   measure of the screened population, not a prevalence estimate, and is
   reported here for comparison with previous rounds of screening only.

The gap between the first two figures is a measurement difference; between the
second and third, a difference in who was measured. Neither indicates an error.
```

Dix lignes, et elles mettent fin à une discussion qui reviendrait sinon chaque trimestre. La dernière phrase est celle qui travaille.

7.8 Quand vous n'avez que des données de routine

C'est-à-dire la plupart du temps. Trois positions honnêtes.

— **Rapportez la charge de cas, non la prévalence.** « 3 700 enfants dépistés, 362 en malnutrition aiguë selon le PB » est une phrase vraie qui soutient l'approvisionnement.

— **Rapportez la tendance, non le niveau.** Une série de routine est bien plus fiable sur le sens que sur le niveau, à condition que la couverture du dépistage soit stable — et dites qu'elle l'est, avec un chiffre.

— **Ancrez à la dernière enquête.** Là où une enquête existe, le rapport routine-enquête de cette année peut être reporté comme hypothèse énoncée, avec sa date. C'est une hypothèse et elle doit être étiquetée comme telle.

Ce qu'il ne faut pas faire est de présenter un taux de dépistage de routine comme une prévalence de population. C'est le mésusage le plus répandu des données de routine dans ce secteur, et il sous-estime toujours.

7.9 La suite

Chaque leçon de ce cours a été une variante d'une seule question. La dernière la pose directement — qu'a-t-on exactement compté, par qui, sur quelle période — et en fait un bloc que l'on attache à tout chiffre produit par le système.

CHAPITRE 8

Qu'a-t-on exactement compté ?

Leçon 8 sur 8 · 75 min

Quatre questions, un bloc de provenance qui y répond, et l'interrogation d'un chiffre menée de la valeur brute au pourcentage publié. La question à laquelle ce cours existe pour permettre de répondre.

8.1 La question

Quelqu'un présente une diapositive — couverture penta3, 78 %. Quelqu'un d'autre demande ce qui a exactement été compté.

Ce n'est ni une question hostile ni une question technique. C'est la question vers laquelle tout le module 2 et tout le module 3 tendaient, et un chiffre qui ne peut pas y répondre est un chiffre indéfendable, si soigneusement qu'il ait été calculé.

8.2 Quatre volets

Qu'a-t-on compté ? L'élément de données, son opérateur d'agrégation, et si une ventilation par catégories a été réduite en sortie.

Où ? Quelles unités d'organisation, à quel niveau, sur quelle version de la hiérarchie.

Quand ? Quelles périodes, par date d'activité ou de saisie, et leur degré de complétude au moment de l'extraction.

Sur quoi ? Le dénominateur, sa source, son année, et combien des unités de rapportage attendues y ont contribué.

Chacun de ces volets a fait une leçon. Ensemble, ils forment le bloc ci-dessous.

8.3 Le bloc de provenance

```
Figure          Penta3 coverage, district, 2024
Value           77.5%

What            Data element PENTA3_DOSES ("Penta 3rd dose administered"),
                aggregation operator SUM over periods and org units.
                No category disaggregation in the extract; the element has
                none configured.
Where           38 facilities at org unit level 4, hierarchy as pinned in
                extracts/2026-07-28/raw/organisationUnits.json. No facility
                was reassigned during 2024.
When            202401 to 202412, by activity period. Analytics tables last
                run 2026-07-27 23:14 UTC, so late entry up to that point is
                included.
Out of what     Surviving infants, MoH catchment estimates 2024, projected
                from the 2015 census at 2.4% annual growth. Denominator
                pro-rated to the months each facility reported: 9,238 of a
                full-year 11,774.
Completeness    349 of 456 facility-months reported (76.5%). August 29%,
                September 45%. The figure is coverage among reporting
                facility-months and is a lower bound on district coverage.
```

```
Extract          extracts/2026-07-28/, manifest.json, script extract.py 1.3
```

Sept lignes et un titre. Cela prend dix minutes la première fois et deux ensuite, car six des sept sortent directement du manifeste et des métadonnées que l'extraction a déjà tirées.

Attachez-le au chiffre, pas au rapport. Une diapositive portant le nombre et une note renvoyant au bloc est ce qui survit à un transfert.

8.4 Interroger un chiffre que vous n'avez pas produit

Le plus souvent, on vous remet un nombre et on vous demande s'il est utilisable. Six questions, dans l'ordre qui élimine le plus de possibilités le plus vite.

1. Élément de données ou indicateur ? Si c'est un indicateur, demandez les expressions de numérateur et de dénominateur. La plupart des désaccords se règlent ici.

2. De quelle année est le dénominateur ? Un numérateur 2024 sur une projection 2019 est fréquent et sous-estime la couverture de ce que représentent neuf ans de croissance.

3. Quel était le taux de rapportage ? En dessous de 90 %, le chiffre est une borne inférieure. Si personne ne le sait, c'est la réponse à la question de savoir s'il est utilisable.

4. Quel niveau d'unité d'organisation ? Un chiffre qui mélange les niveaux compte deux fois ; un taux de couverture par formation n'est pas fiable pour les raisons de bassin qu'a données la leçon 2.

5. La période était-elle close au moment du tirage ? Un mois récent est provisoire, et un chiffre cité d'un tirage effectué trois jours après la fin du mois ne se reproduira pas.

6. À quoi le compare-t-on ? L'essentiel du mésusage est une comparaison — contre une autre mesure, un autre dénominateur, ou une période au taux de rapportage différent.

```python
def interrogate(figure):
    return {
        "type": figure.get("element_or_indicator"),
        "denominator_year": figure.get("denominator_year"),
        "reporting_rate": figure.get("reporting_rate"),
        "org_unit_level": figure.get("level"),
        "period_complete_at_extraction": figure.get("period_closed"),
        "comparable_to": figure.get("comparable_to"),
    }
```

```r
interrogate <- function(figure) {
  figure[c("element_or_indicator", "denominator_year", "reporting_rate",
           "level", "period_closed", "comparable_to")]
}
```

Un blanc dans l'un de ces champs est le constat. Vous ne demandez à personne de se justifier ; vous établissez si le nombre peut porter la décision pour laquelle il est sur le point d'être employé.

8.5 Ce que le système ne peut pas vous dire

Trois choses qu'aucun travail d'API ne récupère, et chacune doit venir d'une personne.

Pourquoi une formation a cessé de rapporter. Le système enregistre qu'elle l'a fait. La raison — un départ, une tablette cassée, un incident de sécurité, un superviseur parti — est dans la tête de quelqu'un et décide si le trou est un problème de données ou de service.

Ce que l'agent de saisie comprenait du champ. Une colonne intitulée « cas » recueille ce qu'une personne croit être un cas. Deux formations peuvent être cohérentes

chacune et vouloir dire des choses différentes, et c'est le problème de définition pour lequel existe le cours sur les indicateurs.

Ce qui s'est passé hors du bassin. Les données de routine voient ce qui est venu au service. La leçon précédente a décomposé exactement ce que cela exclut.

> Le système est un enregistrement de ce qui a été rapporté, non de ce qui s'est passé. Tout chiffre qu'il produit en hérite, et l'énoncer une fois dans un rapport vaut plus que n'importe quelle décimale supplémentaire.

8.6 Ce que ce cours vous laisse

Vous savez lire une extraction DHIS2 comme la base dont elle vient, agréger le long d'une hiérarchie qui bouge, traiter la complétude comme un dénominateur, tirer la même extraction chaque mois avec un manifeste disant ce que vous avez demandé, poser un chiffre de routine à côté d'une estimation d'enquête et expliquer l'écart, et répondre à ce qui a été compté pour tout nombre que le système produit.

8.7 Ce que le module 3 vous laisse

Trois cours. *Conception d'indicateurs et cadre logique* a défini le nombre. *Analyse d'enquêtes, échantillonnage et pondération* lui a mis un intervalle. Celui-ci l'a retracé jusqu'au système qui l'a produit.

Ensemble, ils répondent à la consigne permanente de cette plateforme — toujours énoncer comment un indicateur est calculé, quel est son dénominateur et quelle décision il éclaire — pour les deux types de données dont ce secteur dispose réellement : une enquête et un système de routine.

Le module 4, *Analyses sectorielles*, vient ensuite et est le plus grand de la colonne vertébrale avec six cours. Il applique tout ceci au secteur sur lequel vous rapportez, en commençant par *Analyse nutritionnelle : PCIMA et SMART* — les normes de croissance de l'OMS, le rapport de plausibilité SMART, les seuils de performance Sphere, et l'estimation de la couverture, là où la différence entre le rapport admissions sur charge attendue et la couverture réelle reçoit enfin les vingt-quatre heures qu'elle réclame.

À propos de cette édition

Ce PDF est produit à partir de la même source que la version web de la formation ; les deux ne peuvent donc pas diverger. L'édition en ligne comporte le code exécutable, les jeux de données téléchargeables et les corrections publiées depuis la production de ce fichier.

> Tous les jeux de données cités dans cette formation sont synthétiques ou entièrement anonymisés. Aucun enregistrement ne renvoie à une personne réelle.

Cette formation est publiée en anglais et en français. L'édition française est une adaptation professionnelle reprenant la terminologie employée par l'UNICEF, l'OMS, le PAM, OCHA et les ministères nationaux — et non une traduction littérale.

Consulter et exécuter la formation en ligne sur data-analysis.cassion.dev/fr/cours/routine-data-and-dhis2

Généré le 28 juillet 2026.