Les lignes qui n'ont jamais été écrites

Une ligne absente n'a aucune valeur à avoir manquante. Construire la grille que les données auraient dû remplir, retrouver où sont passées 1 755 lignes de présence, et comprendre pourquoi une grille complète ne prouve pas que tout le monde a rapporté.

Pierre Robentz Cassion

Leçon 6 sur 8

Jointures et restructuration des données de programme — Faire coïncider les pièces

Ce que couvre cette leçon

- Une ligne absente n'a aucune valeur à avoir manquante
- Construire la grille
- Où sont passées les 1 755
- Absent n'est pas absent
- Une grille complète ne prouve pas que tout le monde a rapporté
- Des clés de période qui se trient
- Aligner un registre journalier sur un agrégat mensuel
- La suite

Une ligne absente n'a aucune valeur à avoir manquante

- Tous les contrôles du cours de nettoyage regardaient des valeurs — cases vides, sentinelles, nombres implausibles.

Construire la grille — En Python

```
students = roster["student_id"].drop_duplicates()
school_days = attendance["attendance_date"].drop_duplicates()

grid = pd.MultiIndex.from_product(
    [students, school_days], names=["student_id", "attendance_date"]
).to_frame(index=False)

complete = grid.merge(attendance, on=["student_id", "attendance_date"], how="left")

print(f"grid {len(grid):,}, actual {len(attendance):,}, "
      f"missing {len(grid) - len(attendance):,}")
```

Construire la grille — En R

```r
grid <- tidyr::expand_grid(
  student_id = unique(roster$student_id),
  attendance_date = unique(attendance$attendance_date)
)

complete <- grid |> left_join(attendance, by = c("student_id", "attendance_date"))

cat(sprintf("grid %d, actual %d, missing %d\n",
            nrow(grid), nrow(attendance), nrow(grid) - nrow(attendance)))
```

Construire la grille — En R

```r
complete <- attendance |>
  tidyr::complete(student_id, attendance_date)
```

Où sont passées les 1 755 — En Python

```
missing = complete[complete["present"].isna()]
by_school = (
    missing.merge(roster[["student_id", "school_id"]], on="student_id")
    .groupby("school_id")
    .agg(missing_rows=("present", "size"),
         days=("attendance_date", "nunique"))
)
print(by_school)
```

Où sont passées les 1 755 — En R

```
complete |>
  filter(is.na(present)) |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  summarise(missing_rows = n(), days = n_distinct(attendance_date), .by = school_id)
```

Où sont passées les 1 755

school_id	lignes manquantes	jours
SCH07	915	15
SCH18	840	15

Absent n'est pas absent — En Python

```
naive = complete["present"].fillna(False)
print("attendance rate treating gaps as absences:", naive.mean())
```

Absent n'est pas absent — En R

```
complete |> summarise(rate = mean(coalesce(present, FALSE)))
```

Absent n'est pas absent — En Python

```
open_days = attendance.merge(roster[["student_id", "school_id"]], on="student_id")
school_calendar = open_days[["school_id", "attendance_date"]].drop_duplicates()

grid = (
    roster[["student_id", "school_id"]]
    .merge(school_calendar, on="school_id")
)
print(len(grid), "student-days the schools were actually open")
```

Absent n'est pas absent — En R

```r
school_calendar <- attendance |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  distinct(school_id, attendance_date)

grid <- roster |>
  select(student_id, school_id) |>
  left_join(school_calendar, by = "school_id", relationship = "many-to-many")
```

Absent n'est pas absent

- Construisez la grille par école, pas par district — L'univers des périodes est une propriété de l'unité de rapportage,...

Absent n'est pas absent

Décider si un trou est un zéro, une valeur manquante ou une période qui ne devrait pas exister n'est pas une question technique. C'est l'analyse, et cela a sa place dans le journal avec son motif.

Une grille complète ne prouve pas que tout le monde a rapporté — En Python

```
expected = (
    vax["facility_id"].nunique()
    * vax["period"].nunique()
    * vax["antigen"].nunique()
)
print(expected, "expected rows;", len(vax), "actual")
```

Une grille complète ne prouve pas que tout le monde a rapporté — En R

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

Une grille complète ne prouve pas que tout le monde a rapporté — En Python

```
coverage = (
    vax[vax["report_submitted"]]
    .groupby(["period", "antigen"])
    .agg(doses=("doses_administered", "sum"), target=("target_population", "sum"))
)
reporting = vax.groupby(["period", "antigen"])["report_submitted"].mean()
```

Une grille complète ne prouve pas que tout le monde a rapporté — En R

```r
vax |>
  summarise(
    reporting_rate = mean(report_submitted),
    doses  = sum(doses_administered[report_submitted]),
    target = sum(target_population[report_submitted]),
    .by = c(period, antigen)
  )
```

Des clés de période qui se trient — En Python

```
vax["period"] = pd.to_datetime(vax["period"])
vax["month"] = vax["period"].dt.strftime("%Y-%m")      # 2024-08, sorts correctly
```

Des clés de période qui se trient — En R

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

Des clés de période qui se trient

- Le mois doit porter son année — Une grille de douze mois indexée sur `month` seul fusionne silencieusement août 2023 et…

Aligner un registre journalier sur un agrégat mensuel — En Python

```
monthly = (
    attendance.assign(month=attendance["attendance_date"].dt.strftime("%Y-%m"))
    .merge(roster[["student_id", "school_id"]], on="student_id")
    .groupby(["school_id", "month"])
    .agg(marks=("present", "size"),
         present=("present", "sum"))
    .reset_index()
)
```

Aligner un registre journalier sur un agrégat mensuel — En R

```
monthly <- attendance |>
  mutate(month = format(attendance_date, "%Y-%m")) |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  summarise(marks = n(), present = sum(present %in% TRUE), .by = c(school_id, month))
```

Aligner un registre journalier sur un agrégat mensuel — En Python

```
last = monthly["month"].max()
print(f"excluding partial period {last}")
monthly = monthly[monthly["month"] < last]
```

Aligner un registre journalier sur un agrégat mensuel — En R

```r
monthly <- monthly |> filter(month < max(month))
```

La suite

- Vous disposez maintenant d'un registre agrégé à la même granularité que la remontée mensuelle.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/joining-and-reshaping/joining-06-time-and-periods