CSV, Excel et largeur fixe, lus sans dommage

Encodages, séparateurs, classeurs multi-feuilles, lignes d'en-tête fusionnées et spécifications de colonnes — les pièges propres à chaque format, qui corrompent un fichier avant même que vous n'ayez regardé une seule valeur.

Pierre Robentz Cassion

Leçon 3 sur 8

Python pour les données de programme — Faire entrer l'export

Ce que couvre cette leçon

- Ce que cette leçon suppose
- Le CSV n'est pas un format unique
- Excel
- Fichiers à largeur fixe
- Vérifier la lecture avant d'en faire quoi que ce soit
- Lire un gros fichier
- Ce qui vient ensuite

Ce que cette leçon suppose

- *Fondamentaux de l'analyse de données* couvre le principe : déclarez vos types, déclarez vos sentinelles, et ne laissez jamais un lecteur en inférer l'un ou l'autre.

Le CSV n'est pas un format unique

- Le séparateur

Le CSV n'est pas un format unique — En Python

```python
import pandas as pd

# Faux : tout atterrit dans une colonne unique nommee d'apres l'en-tete entier.
wrong = pd.read_csv("export.csv")
print(wrong.shape)              # (2736, 1)

right = pd.read_csv("export.csv", sep=";", decimal=",")
print(right.shape)              # (2736, 7)
```

Le CSV n'est pas un format unique

- L'encodage

Le CSV n'est pas un format unique — En Python

```
# UnicodeDecodeError, ou pire, silencieusement abime : "Gona\xefves"
muac = pd.read_csv(path)

muac = pd.read_csv(path, encoding="utf-8")       # ce que vous voulez
muac = pd.read_csv(path, encoding="latin-1")     # ce qu'Excel produit souvent
```

Le CSV n'est pas un format unique

- Les séparateurs de milliers transforment les nombres en texte

Le CSV n'est pas un format unique — En Python

```
# target_population arrive en "1,480" et pandas le lit comme une chaine.
epi = pd.read_csv(path, thousands=",")
```

Le CSV n'est pas un format unique

- Les zéros initiaux

Le CSV n'est pas un format unique — En Python

```
# facility_id "007" devient l'entier 7 ; la jointure a la liste des formations
# echoue exactement pour celles dont le code portait un zero initial.
epi = pd.read_csv(path)

epi = pd.read_csv(path, dtype={"facility_id": "string"})
```

Le CSV n'est pas un format unique — En Python

```
# Tout lire en texte d'abord, regarder, puis convertir deliberement.
peek = pd.read_csv(path, dtype="string", nrows=200)
print(peek.head())
print(peek.columns.tolist())
```

Excel

- Il y a plus d'une feuille

Excel — En Python

```
book = pd.ExcelFile(path)
print(book.sheet_names)
# ['Cover', 'Data', 'Codebook', 'Sheet3']

data = pd.read_excel(path, sheet_name="Data")
```

Excel — En Python

```
sheets = pd.read_excel(path, sheet_name=None)
monthly = pd.concat(sheets, names=["sheet"]).reset_index(level="sheet")
```

Excel

- L'en-tête n'est pas en ligne 1

Excel — En Python

```
data = pd.read_excel(path, sheet_name="Data", skiprows=3)
```

Excel

- Les cellules fusionnées produisent des valeurs manquantes

Excel — En Python

```
data["district"] = data["district"].ffill()
```

Excel

- Les dates Excel

Excel — En Python

```
# 45306 correspond au 15/01/2024
data["screening_date"] = pd.to_datetime(
    data["screening_date"], unit="D", origin="1899-12-30"
)
```

Fichiers à largeur fixe — Exemple

```
CH00854Terre-Neuve    2024-01-15022m133false
CH01207Gonaives       2024-01-15034f118false
```

Fichiers à largeur fixe — En Python

```
COLSPECS = [(0, 7), (7, 22), (22, 32), (32, 35), (35, 36), (36, 39), (39, 44)]
NAMES = ["child_id", "commune", "screening_date", "age_months", "sex",
         "muac_mm", "oedema"]

muac = pd.read_fwf(
    path,
    colspecs=COLSPECS,
    names=NAMES,
    dtype={"child_id": "string", "commune": "string"},
)

muac["commune"] = muac["commune"].str.strip()
```

Fichiers à largeur fixe

- **Les plages sont semi-ouvertes**, comme les tranches Python : `(0, 7)` couvre les caractères 0 à 6. Une erreur d'un…
- **Retirez le remplissage.** Les champs à largeur fixe sont complétés par des espaces : `"Terre-Neuve "` ne sera pas…
- **Prenez la spécification dans la documentation, pas en comptant à l'écran.** `read_fwf` sait inférer `colspecs`, et il…

Vérifier la lecture avant d'en faire quoi que ce soit — En Python

```
def check(df, expected_rows=None):
    print("forme     ", df.shape)
    print("types     ", df.dtypes.value_counts().to_dict())
    print("manquants ", df.isna().sum().to_dict())
    print("1re ligne ", df.iloc[0].to_dict())
    if expected_rows is not None:
        assert len(df) == expected_rows, f"attendu {expected_rows}, obtenu {len(df)}"

muac = pd.read_csv(
    RAW / "muac-screening-artibonite-2024.v1.csv",
    dtype={"child_id": "string", "commune": "string", "sex": "string"},
    na_values={"muac_mm": ["-99"]},
)
check(muac, expected_rows=4218)
```

Vérifier la lecture avant d'en faire quoi que ce soit

- **la forme** attrape le mauvais séparateur et un téléchargement tronqué.
- **les types** attrapent les zéros initiaux perdus, les séparateurs de milliers et une colonne numérique lue comme du...
- **les manquants** attrapent un mauvais encodage et une sentinelle non déclarée.
- **la première ligne** attrape un décalage d'en-tête — si elle ressemble à un en-tête, c'en était un.

Lire un gros fichier — En Python

```
COLUMNS = ["student_id", "attendance_date", "present"]
attendance = pd.read_csv(path, usecols=COLUMNS, dtype={"student_id": "string"})
```

Lire un gros fichier — En Python

```
totals = []
for chunk in pd.read_csv(path, chunksize=100_000, dtype={"student_id": "string"}):
    totals.append(chunk.groupby("student_id")["present"].sum())

per_student = pd.concat(totals).groupby(level=0).sum()
```

Ce qui vient ensuite

- Le fichier est en mémoire, forme et types intacts.

Lire la leçon complète, avec le code exécutable

https://data-analysis.cassion.dev/fr/cours/python-for-programme-data/python-03-reading-exports