CSV, Excel and fixed-width, read without damage

Encodings, separators, multi-sheet workbooks, merged header rows and column specifications — the format-specific traps that corrupt a file before you have looked at a single value.

Pierre Robentz Cassion

Lesson 3 of 8

Python for Programme Data — Getting the export in

What this lesson covers

- What this lesson assumes
- CSV is not one format
- Excel
- Fixed-width files
- Verify the read before doing anything with it
- Reading a large file
- What comes next

What this lesson assumes

- *Data Analysis Foundations* covers the principle: declare your types, declare your sentinels, and never let a reader infer either.

CSV is not one format

- The separator

CSV is not one format — In Python

```
import pandas as pd

# Wrong: everything lands in one column named after the whole header line.
wrong = pd.read_csv("export.csv")
print(wrong.shape)              # (2736, 1)

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

CSV is not one format

- The encoding

CSV is not one format — In Python

```
# UnicodeDecodeError, or worse, silently mangled: "Gona\xefves"
muac = pd.read_csv(path)

muac = pd.read_csv(path, encoding="utf-8")       # what you want
muac = pd.read_csv(path, encoding="latin-1")     # what Excel often produces
```

CSV is not one format

- Thousands separators turn numbers into text

CSV is not one format — In Python

```
# target_population arrives as "1,480" and pandas reads it as a string.
epi = pd.read_csv(path, thousands=",")
```

CSV is not one format

- Leading zeros

CSV is not one format — In Python

```
# facility_id "007" becomes the integer 7; the join to the facility list fails
# for exactly the facilities whose code had a leading zero.
epi = pd.read_csv(path)

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

CSV is not one format — In Python

```
# Read everything as text first, look, then convert deliberately.
peek = pd.read_csv(path, dtype="string", nrows=200)
print(peek.head())
print(peek.columns.tolist())
```

Excel

- There is more than one sheet

Excel — In Python

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

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

Excel — In Python

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

Excel

- The header is not row 1

Excel — In Python

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

Excel

- Merged cells produce missing values

Excel — In Python

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

Excel

- Excel dates

Excel — In Python

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

Fixed-width files — Example

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

Fixed-width files — In 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()
```

Fixed-width files

- **Ranges are half-open**, like Python slices: `(0, 7)` is characters 0 to 6. Off-by-one here shifts every subsequent…
- **Strip the padding.** Fixed-width fields are space-padded, so `"Terre-Neuve "` will not compare equal to…
- **Get the spec from the documentation, not by counting on screen.** `read_fwf` can infer `colspecs`, and it infers them…

Verify the read before doing anything with it — In Python

```
def check(df, expected_rows=None):
    print("shape     ", df.shape)
    print("dtypes    ", df.dtypes.value_counts().to_dict())
    print("missing   ", df.isna().sum().to_dict())
    print("first row ", df.iloc[0].to_dict())
    if expected_rows is not None:
        assert len(df) == expected_rows, f"expected {expected_rows}, got {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)
```

Verify the read before doing anything with it

- **shape** catches the wrong separator and a truncated download.
- **dtypes** catches leading zeros lost, thousands separators, and a numeric column read as text.
- **missing** catches a wrong encoding and a sentinel not declared.
- **first row** catches a header offset — if the first row looks like a header, it was one.

Reading a large file — In Python

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

Reading a large file — In 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()
```

What comes next

- The file is in memory with its shape and types intact.

Read the full lesson, with runnable code

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