Codes, sentinels and the 99 that becomes a mean

Missing-value sentinels, categorical vocabularies, booleans recorded five ways, and Stata value labels — turning what the form recorded into what pandas can compute on.

Pierre Robentz Cassion

Lesson 4 of 8

Python for Programme Data — Getting the export in

What this lesson covers

- A number that means "no answer"
- Categorical vocabularies
- Booleans recorded five ways
- Text that should be one value
- Stata and SPSS carry their labels with them
- The read function, assembled
- What comes next

A number that means "no answer" — In Python

```
import pandas as pd

muac = pd.read_csv(RAW / "muac-screening-artibonite-2024.v1.csv")
print(muac["muac_mm"].mean())        # several millimetres below the truth
```

A number that means "no answer"

- Declare sentinels at read time

A number that means "no answer" — In Python

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

print(muac["muac_mm"].mean())
print(muac["muac_mm"].isna().sum())
```

A number that means "no answer"

- Scope the sentinel to the column — `na_values=["-99"]` without a dictionary applies it to every column in the file,...
- The sentinel list is documentation, not folklore

A number that means "no answer" — In Python

```
SENTINELS = {
    "muac_mm": ["-99"],           # not measured
    "age_months": ["99", "-1"],   # not answered / not applicable
}

muac = pd.read_csv(path, na_values=SENTINELS)
```

A number that means "no answer" — In Python

```
for column in ["muac_mm", "age_months"]:
    print(column, sorted(muac[column].dropna().unique())[:5],
          sorted(muac[column].dropna().unique())[-5:])
```

A number that means "no answer"

Declaring a sentinel is not deciding what to do about the missing value. That decision — drop, impute, report separately — belongs to the analysis, and it has to be written down. Here you are only stopping a code from pretending to be a measurement.

Categorical vocabularies — In Python

```
OUTCOMES = pd.CategoricalDtype(
    ["no-action", "referred-tsfp", "referred-otp", "referred-sc"], ordered=False
)

muac["outcome"] = muac["outcome"].astype(OUTCOMES)
print(muac["outcome"].isna().sum())
```

Categorical vocabularies

- Count the missing immediately after converting — This is the trap: a value outside the declared categories becomes...

Categorical vocabularies — In Python

```
raw_values = set(muac_raw["outcome"].dropna().unique())
declared = set(OUTCOMES.categories)
print("unexpected:", raw_values - declared)
```

Categorical vocabularies

- Ordered categories

Categorical vocabularies — In Python

```
LADDER = pd.CategoricalDtype(
    ["surface-water", "unimproved", "limited", "basic", "safely-managed"],
    ordered=True,
)

wash["service"] = wash["service"].astype(LADDER)
at_least_basic = wash["service"] >= "basic"
```

Booleans recorded five ways — In Python

```
print(muac["oedema"].value_counts(dropna=False))
```

Booleans recorded five ways — In Python

```
# Wrong in a way that is hard to see: every non-empty string is truthy,
# so "false" becomes True.
muac["oedema"] = muac["oedema"].astype(bool)
```

Booleans recorded five ways — In Python

```
BOOLEANS = {
    "true": True, "TRUE": True, "Y": True, "y": True, "yes": True, "1": True,
    "false": False, "FALSE": False, "N": False, "n": False, "no": False, "0": False,
}

muac["oedema"] = (
    muac["oedema"].astype("string").str.strip().str.lower()
    .map({k.lower(): v for k, v in BOOLEANS.items()})
)

print(muac["oedema"].isna().sum())    # anything the map did not cover
```

Booleans recorded five ways — In Python

```
muac["oedema"] = muac["oedema"].astype("boolean")      # True / False / <NA>
```

Text that should be one value — In Python

```
print(wash["district"].value_counts())
# Nord-Ouest     727
# NORD-OUEST      33
# Nord Ouest      22
# nord-ouest      20
```

Text that should be one value — In Python

```
wash["district"] = (
    wash["district"].astype("string").str.strip().str.lower()
    .str.replace(r"\s+", "-", regex=True)
)
print(wash["district"].nunique())     # 3
```

Text that should be one value — In Python

```
CANONICAL = {
    "gonaives": "Gonaïves",
    "st-marc": "Saint-Marc",
    "saint-marc": "Saint-Marc",
}
wash["district"] = wash["district"].map(CANONICAL).fillna(wash["district"])
```

Stata and SPSS carry their labels with them — In Python

```
survey = pd.read_stata(path)                          # values converted to labels
survey = pd.read_stata(path, convert_categoricals=False)   # raw codes
```

Stata and SPSS carry their labels with them — In Python

```
with pd.io.stata.StataReader(path) as reader:
    labels = reader.value_labels()
    variables = reader.variable_labels()

print(variables["hh_size"])
print(labels.get("consent", {}))
```

The read function, assembled — In Python (cont.)

```
from pathlib import Path
import pandas as pd

SENTINELS = {"muac_mm": ["-99"]}
OUTCOMES = pd.CategoricalDtype(
    ["no-action", "referred-tsfp", "referred-otp", "referred-sc"]
)
BOOLEANS = {"true": True, "y": True, "yes": True, "1": True,
            "false": False, "n": False, "no": False, "0": False}

def read_register(path: Path) -> pd.DataFrame:
    muac = pd.read_csv(
        path,
        dtype={"child_id": "string", "commune": "string", "sex": "string"},
        na_values=SENTINELS,
    )
```

The read function, assembled — In Python (cont.)

```
    muac["screening_date"] = pd.to_datetime(
        muac["screening_date"], format="%Y-%m-%d", errors="raise"
    )
    muac["outcome"] = muac["outcome"].astype(OUTCOMES)
    muac["oedema"] = (
        muac["oedema"].astype("string").str.strip().str.lower()
        .map(BOOLEANS).astype("boolean")
    )

    assert muac["child_id"].notna().all(), "every row needs an identifier"
    return muac
```

What comes next

- The table is in memory and every column means what it says.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/python-for-programme-data/python-04-codes-and-sentinels