SLUB
Wir führen Wissen.

Metadata for Pandas

Read and Write FAIR Tabular Data with Pythons Pandas Library

Arne Rümmler, SLUB Dresden

Problem Breakdown

- We want to foster the creation of FAIR research data
- Pythons Pandas library is widely used in research
- CSV is a common exchange format for tabular data
- **CSV is inherently unFAIR** (by the means of Interoperability and Reusability)

Excerpt from FAIR-Principles:

I1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation

R1: (Meta)data are richly described with a plurality of accurate and relevant attributes

Problem 1: Interoperability

CSV is not formally defined

*It has been suggested that the **"V"** in the acronym **stands for "Vague"** instead of "Values". Different delimiters and quoting characters are just the start.*
PEP 305 – CSV File API

german_cities.csv

```
City     Population [10^6]    Area [km^2] Postal Code
Berlin   3,664    891,85   10115
Hamburg  1,899    755,16   20095
Frankfurt    0,732    248,31   60306
Dresden  0,549    328,31   01067
Aachen   0,246    160,85   52062
```

```python
df = pd.read_csv("german_cities.csv")
df.head()
```

		City\tPopulation [10^6]\tArea [km^2]\tPostal Code
Berlin\t3	664\t891	85\t10115
Hamburg\t1	899\t755	16\t20095
Frankfurt\t0	732\t248	31\t60306
Dresden\t0	549\t328	31\t01067
Aachen\t0	246\t160	85\t52062

```python
df = pd.read_csv(
    "german_cities.csv",
    sep="\t",
    decimal=",",
    dtype={"Postal Code": str},
)
df.head()
```

	City	Population [10^6]	Area [km^2]	Postal Code
0	Berlin	3.664	891.85	10115
1	Hamburg	1.899	755.16	20095
2	Frankfurt	0.732	248.31	60306
3	Dresden	0.549	328.31	01067
4	Aachen	0.246	160.85	52062

Problem 1: Interoperability

CSV is not formally defined

Most CSV parsers assume that the file adheres to RFC 4180.

RFC 4180 CSV Format Definition

- 0 or 1 header row
- each row after the header is a record (no *totals* row etc.)
- field separator: comma
- quotation: double quotes
- all rows have the same number of fields
- line break: \r\n (CRLF)

→ **not used universally**
→ **doesn't cover all edge cases**

```
(function) def read_csv(
    sep
    delimiter
    header
    index_col
    usecols
    dtype
    true_values
    false_values
    skipinitialspace
    skiprows
    skipfooter
    na_values
    skip_blank_lines
    thousands
    decimal
    lineterminator
    quotechar
    doublequote
    escapechar
    comment
    encoding
    ...
)
```

The format properties of a CSV file are called its **dialect**.

Problem 2: Reusability

CSV has no formal way to express semantics

Required:

- table-wide descriptive metadata
- exhaustive description of the columns (datatype, data format information (date, decimal separator), units, …)

Situation: different approaches, e.g. preamble, comments, …

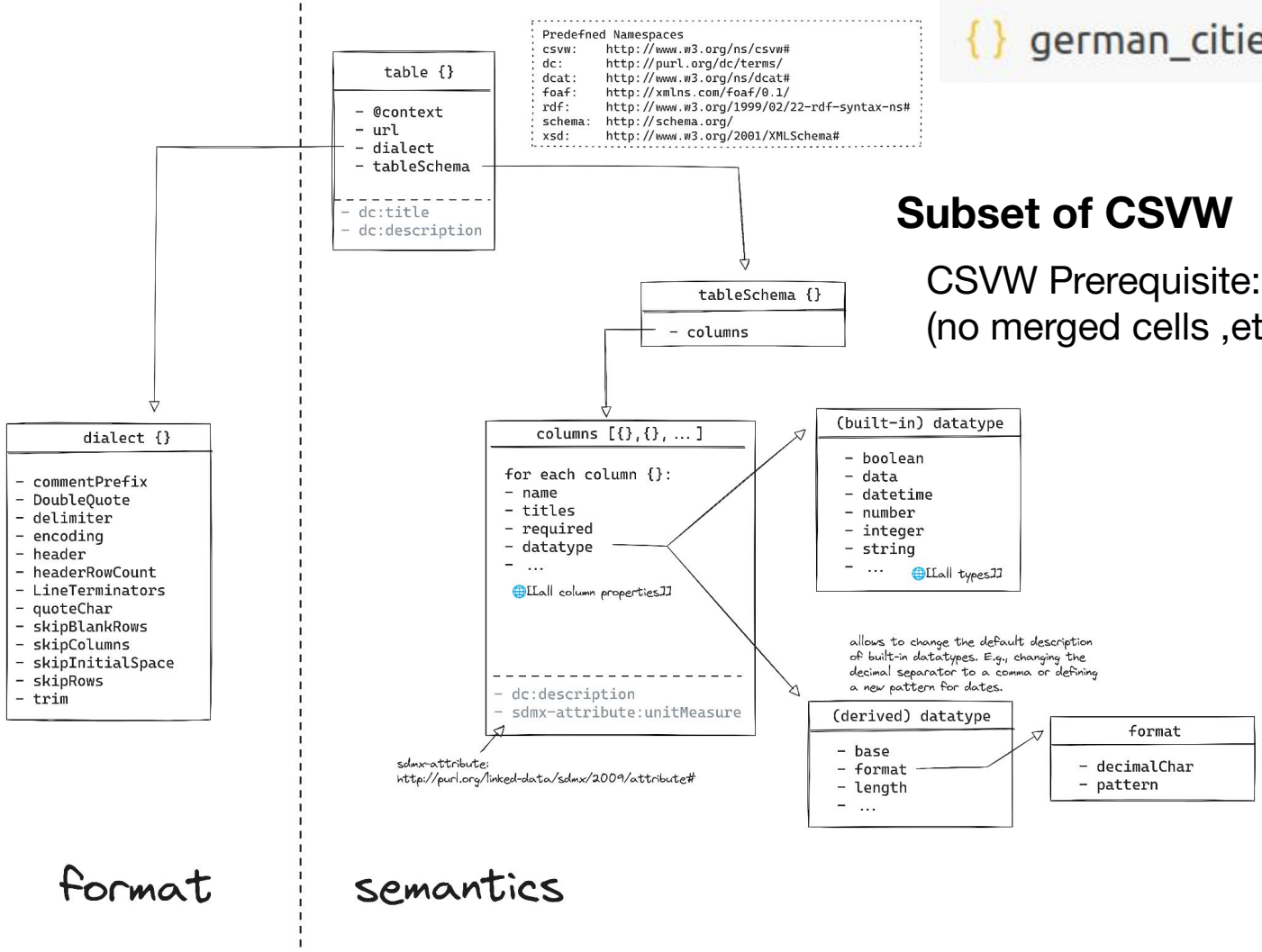
```
# @description: the table contains information about a selection of German cities.
# @author: Arne Rümmler
# @separator: \t
#
City    Population  Area    Postal Code
# @datatype: string decimal decimal string
# @unit: -  million square kilometres   -
Berlin  3,664   891,85  10115
Hamburg 1,899   755,16  20095
Frankfurt   0,732   248,31  60306
Dresden 0,549   328,31  01067
Aachen  0,246   160,85  52062
```

CSVW - A W3C Recommendation

Add a JSON Metadata File

german_cities.csv

german_cities.csv-metadata.json



Subset of CSVW

CSVW Prerequisite: “rectangular” table (no merged cells ,etc)

Questions so far?

Wishlist:

- derive class from `pandas.DataFrame()` -> `MetaDataFrame()`
- override class method `pandas.DataFrame.to_csv()` to write a `*.csv-metadata.json` along the `*.csv`
- override `pandas.read_csv()` to seach for and parse `*.csv-metadata.json`

Today:

- `def read_csv_metadata(path: str): pd.DataFrame`
- `def write_csv_metadata(df: pd.DataFrame, ...): None`

https://github.com/rue-a/csvw-workshop

Links

Ressources

- CSVW overview docs: https://csvw.org/
- CSVW documents: https://www.w3.org/2013/csvw/wiki/Main_Page.html

Similar Projects/Tools (non-exhaustive)

- Python CSVW package (read/write/validate csv(w) and convert to frictionless data package): https://github.com/cldf/csvw
- R-language data.frame metadata: https://dataset.dataobservatory.eu/
- CSV-X (describe any csv, regardless of its weirdness): https://ieeexplore.ieee.org/document/7917195
- Frictionlessdata.io supports CSV-Dialect: https://specs.frictionlessdata.io/csv-dialect/#language

The End.

Please direct upcoming questions towards:

arne.ruemmler@slub-dresden.de or

openscience@slub-dresden.de