

Building CSV-powered tools for social sciences

csv,conf,v9 2025 in Bologna

Guillaume Plique (aka @Yomguithereal)
médialab SciencesPo, Paris

The médialab

The **médialab** is a social sciences research lab gathering:

- Researchers
- Engineers
- Designers

We have been building **Open-Source** tools geared towards social sciences & civil society at large for 10 years now!

Check out: **<https://github.com/medialab/>** (<https://github.com/medialab/>)

A CSV-shaped question

■ Most of our tools/research use some kind of CSV data

We love CSV so much we wrote a ♥Love Letter♥ !

https://github.com/medialab/xan/blob/master/docs/LOVE_LETTER.md

■ Why though?

Because CSV is an affordable, understandable and free data format.

So much so that CSV is a data *lingua franca* for our research engineering work.

Affordances of tabular data (1/2)

Everybody is familiar with tabular data, at least through spreadsheets:

name	surname	age
Guillaume	Plique	35
Lucy	Miller	37
Marina	Spring	45

Affordances of tabular data (2/2)

Less so with nested or hierarchical data structures like the ones found in JSON:

```
{
  "people": [
    {
      "name": "Guillaume",
      "surname": "Plique",
      "age": 35,
      "friends": ["lucy", "marina"]
    }
  ]
}
```

CSV is simple

The specification holds in the title: **C**omma **S**eparates **V**alues!

(Everyone here knows it's a lie, sure you need to quote values with newlines and double your quotes, but you see the point...)

In any case: it can be explained to anybody in mere **minutes**.

Even more so: the format is so simple you might invent it **yourself**.

(Or should I say **discover** it)

CSV is free (1/2)

Nobody owns the CSV format.

It is and will always be a free and open collective idea.

It is so free it has **no real specification** either.

(Don't force me to rant about RFC 4180, pretty please)

It is so free some disciplines have their own CSV **dialects**:

- bioinformatics: **.vcf**, **.gff**, **.gtf** etc.
- web archives: **.cdx** etc.

CSV is free (2/2)

You don't need **proprietary** software to process/read/write CSV data.

As a cheeky counterpoint, let's open an Excel file:

```
vim data.xls
```

[illegible]

CSV is just text

You can **write** it yourself, to some degree.

You can **read** it yourself, to some degree.

If I hand out some CSV to you, you should be able to understand what's going on very easily in this piece of text.

And you will probably still be able to do so it in 50 years.

CSV is a bridge

CSV can be handled by both **researchers**, **students** and **engineers** alike.

CSV is a bridge between the **spreadsheet** world and the **engineering** world.

CSV is a good fit for both **tiny** & **big** data problem.

CSV creates the perfect conditions for the de facto auto-organized **interoperability** of a lot of free tools designed and maintained by many people around the world!

An ethos of sobriety

Although not exactly a chosen one ;)

- We don't have access to powerful hardware.
 - Most of our users (students, researchers, civil society) don't have access to powerful hardware neither.
-

What makes CSV a sober data format?

CSV is naturally succinct

1. Headers are only written once
2. Strings are optimally represented (except for quotes)
3. Numbers could be lighter, but not by much

CSV is efficient

Can you reliably collect billions of tweets without databases, relying only on gigabytes of CSV data?

Sure! (At least this is what we did for years before the evil dude arrived)

CSV can be thought of as a structured append-only log format.

Building tools around CSV (1/5)

■ Dedicated web UIs

Table 2 Net

⬇ Metadata Tools




Table 2 Net

Extract a network from a table. Set a column for nodes and a column for edges. It deals with multiple items per cell.

Load your CSV table

It has to be **comma-separated** and the first row must be dedicated to **column names**.

Parsing successful. 4 columns and 11 rows.

Table preview




Row number	source	target	weight
1	A	B	0.5
2	B	A	1
3	B	C	100
4	D	E	0.7
5	A	C	1.9
6	B	B	12
7	C	A	0.6
8	E	F	15.8

1. Type of Network

Choose type of network...


You may extract different types of networks from a table. It depends on how you use columns to build the nodes and the edges.





- Normal:** If you want a single type of nodes, for instance authors. They will be linked when they share a value in another column, for instance papers.
- Bipartite:** If you want two types of nodes, for instance authors and papers, they will be linked when they appear in the same row of the table.
- Citation:** If you have a column containing references to another one, for instance paper title and cited papers (title).



Building tools around CSV (2/5)

■ Dedicated web UIs

**Takoyaki**



Fingerprint Collision

Computing clusters of strings whose fingerprints are the same. This is definitely the first recipe you want to try since it is really performant and usually produces very good results.

Running this recipe requires ~200 computations

Cologne Collision

This recipe will compute clusters of strings having the same Cologne phonetic code. While this seems to be useful only for Chinese names, it actually tend to work quite well on large arrays of strings.

Running this recipe requires ~200 computations

Metaphone Collision

This recipe will compute clusters of strings having the same metaphone encoding, i.e. strings that are pronounced roughly the same.

Running this recipe requires ~200 computations

My custom recipe

Description of my custom clustering recipe

Running this recipe requires ~200 computations

[edit choice](#)

Levenshtein VPTree

This recipe leverages a merge-sort tree to perform a clustering using the Levenshtein distance.

Running this recipe requires ~20 computations

Levenshtein Naïve

This recipe will try to gather similar strings, i.e. strings whose Levenshtein distance is below a given threshold (usually <= 2).

Running this recipe requires ~400 computations

Create a custom recipe

Found 40 clusters in the Publisher column using the **Fingerprint Collision** recipe.

Sort clusters by: **number ↑**, **distinct values**, **affected rows**

387	Elsevier	x	Harmonize
8	Elsevier	x	Drop
4	ELSEVIER	x	Explore
↓	Elsevier		
14	American Chemical Society	x	Harmonize
1	AMERICAN CHEMICAL SOCIETY	x	Drop
↓	American Chemical Society		Explore
7	International Union of Crystallography	x	Harmonize
1	International Union of Crystallography	x	Drop
↓	International Union of Crystallography		Explore
81	Springer	x	Harmonize
1	Springer	x	Drop
↓	Springer		Explore
56	Wiley-Blackwell	x	Harmonize
1	Wiley/Blackwell	x	Drop
↓	Wiley-Blackwell		Explore

Back

Re-cluster

Building tools around CSV (3/5)

■ Command line tools

<https://github.com/medialab/minet>

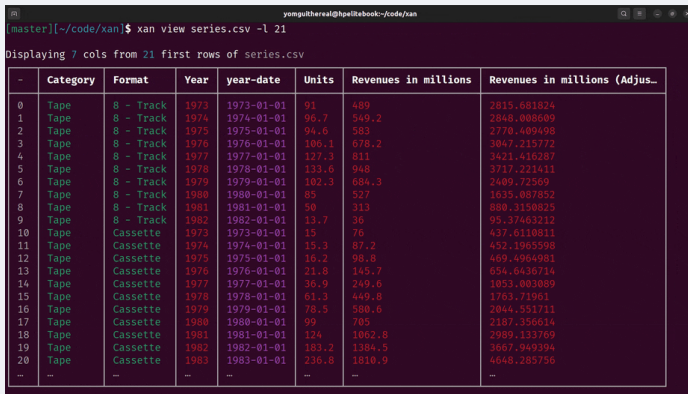
```
(minet) *[master][~/code/minet]$ minet fetch url -i ftest/resources/urls.csv -o report.csv --resume
```

CSV in, CSV out

Building tools around CSV (4/5)

■ Command line tools

<https://github.com/medialab/xan>



A terminal window titled 'yomguthereal@hpelirebook: ~/code/xan' shows the command `xan view series.csv -l 21` being executed. The output indicates that 7 columns from the first 21 rows of 'series.csv' are being displayed. The data is presented in a table with 8 columns: index, Category, Format, Year, year-date, Units, Revenues in millions, and Revenues in millions (Adjus...).

-	Category	Format	Year	year-date	Units	Revenues in millions	Revenues in millions (Adjus...
0	Tape	8 - Track	1973	1973-01-01	91	489	2815.681824
1	Tape	8 - Track	1974	1974-01-01	96.7	549.2	2848.008609
2	Tape	8 - Track	1975	1975-01-01	94.6	583	2770.409498
3	Tape	8 - Track	1976	1976-01-01	106.1	678.2	3047.215772
4	Tape	8 - Track	1977	1977-01-01	127.3	811	3421.416287
5	Tape	8 - Track	1978	1978-01-01	133.6	948	3717.221411
6	Tape	8 - Track	1979	1979-01-01	102.3	684.3	2409.72569
7	Tape	8 - Track	1980	1980-01-01	85	527	1635.087852
8	Tape	8 - Track	1981	1981-01-01	50	313	880.3150825
9	Tape	8 - Track	1982	1982-01-01	13.7	36	95.37463212
10	Tape	Cassette	1973	1973-01-01	15	76	437.6110811
11	Tape	Cassette	1974	1974-01-01	15.3	87.2	452.1965598
12	Tape	Cassette	1975	1975-01-01	16.2	98.8	469.4964981
13	Tape	Cassette	1976	1976-01-01	21.8	145.7	654.6436714
14	Tape	Cassette	1977	1977-01-01	36.9	249.6	1053.003089
15	Tape	Cassette	1978	1978-01-01	61.3	449.8	1763.71961
16	Tape	Cassette	1979	1979-01-01	78.5	580.6	2044.551711
17	Tape	Cassette	1980	1980-01-01	99	705	2187.356614
18	Tape	Cassette	1981	1981-01-01	124	1062.8	2989.133769
19	Tape	Cassette	1982	1982-01-01	183.2	1384.5	3667.949394
20	Tape	Cassette	1983	1983-01-01	236.8	1810.9	4648.285756
...

Building tools around CSV (5/5)

■ Libraries

<https://github.com/medialab/casanova>

```
import casanova

with open("cities.csv") as input_file \
    open("enriched-cities.csv") as output_file:

    enricher = casanova.enricher(
        input_file, output_file,
        add=["pop_ratio"]
    )

    for row in enricher:
        ...
```

CSV rewards out-of-the-box engineering

■ Reading CSV data in reverse

Thanks to backslash-free quote escaping!

```
# Constant memory!  
# Amortized linear time!  
xan reverse file.csv  
xan tail file.csv
```

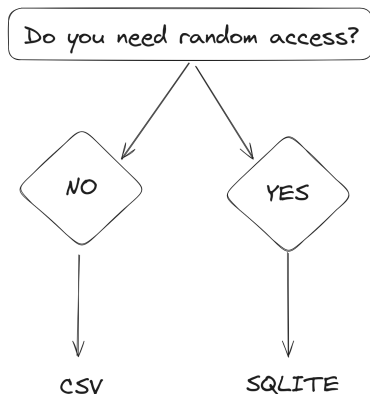
■ Resuming

Thanks to reverse reading!

```
minet fetch url -i posts.csv --resume -o report.csv
```

Conclusion: always bet on CSV!

Choosing a database: a flowchart



Thank you for your time !

Some CSV-relevant links:

- **médialab's Open Source**: <https://github.com/medialab/>
- **Love Letter**: https://github.com/medialab/xan/blob/master/docs/LOVE_LETTER.md
- **xan**: <https://github.com/medialab/xan>
- **minet**: <https://github.com/medialab/minet>
- **casanova**: <https://github.com/medialab/casanova>
- **table2net**: <https://medialab.github.io/table2net/>
- **takoyaki**: <https://yomguithereal.github.io/takoyaki/>
- **pelote**: <https://github.com/medialab/pelote/>