

PREPRINT

HistWarDB: building a historical war database out of Wikidata

Vincenzo Manto, Riccardo Dal Cero

October 8, 2026

DOI: <https://doi.org/10.5281/zenodo.23243673>

<https://github.com/VincenzoManto/histwardb>

Abstract

HistoricalWarDB is a dynamic, programmatic Python toolkit designed to extract, clean, and analyze global warfare data from Wikidata spanning from 1700 to the present. Built for researchers, historians, and data scientists, it bridges the gap between crowdsourced knowledge and rigorous historical data pipelines, supporting both modern sovereign states and historical empires.

Keywords: Data Engineering, History, Open Source, Political Science

A while back, a friend of mine, [Riccardo Dal Cero](#), asked me something that sounded deceptively simple: did a dataset exist that, for every country and year, told you whether that country was at war? He needed it for a historical/political analysis — something clean, tabular, that he could work with without having to reconstruct everything by hand from Wikipedia or scattered sources.

The short answer was: I don't know. There are academic datasets (COW, UCDP), but they're often partially paywalled, awkward to normalize, limited in temporal coverage, or licensed in unclear ways. The long answer became **HistWarDB**.

The problem

Wikidata already holds a huge amount of information about historical conflicts: typed entities, relations, start and end dates, participants. The problem is that it's a massive, noisy graph: peacekeeping missions labeled ambiguously, military exercises that pop up as "conflicts," historical states that change names or die and are reborn under different entities (the German Empire, the Qing dynasty, the Kingdom of Montenegro...). Querying Wikidata via SPARQL over a multi-century range, on top of that, means running straight into the public endpoint's query limits.

So the job wasn't "download a dataset" — it was building a pipeline that would:

1. Reliably query Wikidata across a 326-year span (1700–2026).
2. Filter out noise (peacekeeping missions, military exercises, non-state actors) without throwing away real signal.
3. Correctly handle historical entities, with chronological sanity checks on their lifespans.
4. Return the data in different shapes depending on the use case: long format for statistical analysis, binary matrices for heatmaps, and so on.

The architecture

I split the project into three independently testable layers — a deliberate choice meant to allow a full test suite without ever having to hit live Wikidata:

- ``query.py`` — pure functions that build the SPARQL query text, splitting the time range into 25-year chunks to stay under Wikidata's limits.
- ``core.py` (fetch)` — the network layer, with an injectable `requests`-like session: tests swap in a fake session instead of touching the real network.
- ``core.py` (clean)` — parsing, anti-noise filtering (excluding `wd:Q35349` for peacekeeping, `wd:Q1151430` for military exercises, non-state actors), anachronism checks, and construction of the country $\times$ year grid.

This separation of I/O, query construction, and cleaning logic is what makes the whole thing reproducible: given the same Wikidata snapshot, the pipeline always returns the same output, and it's all verifiable with `pytest` without depending on the live service being up.

Using it

The goal was that going from zero to a heatmap would take four lines:

```
from histwardb import HistWarDB

# Initialize the dataset from 1700 to 2026
gwd = HistWarDB(start_year=1700, end_year=2026)

# Query specific countries in long format
france_uk_wars = gwd(countries=["France", "United Kingdom"], view="long",
only_in_war=True)

# Plot an interactive heatmap for the top 15 conflict-prone states
gwd.plot_heatmap(top_n=15, from_year=1800, to_year=2026).show()
```

Every view (`long`, `binary`, `names`, `list`, `countries`, `conflicts`) respects the same filters — `countries`, `date range`, `only_in_war`, `keyword` — so you can move from an aggregate analysis to a binary matrix ready for plotting without rewriting the query.

View	Returns
<code>`long`</code>	Tidy DataFrame: Country, Year, War_State, Active_Conflicts
<code>`binary`</code>	Wide pivot: countries $\times$ years, 1 = at war, 0 = peace
<code>`names`</code>	Wide pivot with literal conflict names active per year
<code>`list`</code>	Alphabetical list of unique conflict names under current filters
<code>`countries`</code>	Sorted list of active states matching the query
<code>`conflicts`</code>	Sorted list of unique conflicts matching the query

What came out of it

The result is a dataset with **326 years** of coverage, built from 25-year SPARQL chunks, **6 analytical views**, an MIT license, and a full test suite that doesn't require live access to Wikidata to run. It's open source, published on [GitHub](#), with an interactive demo available [here](#).

What started as a favor — "I need a country/year/war CSV" — turned into a small data engineering project with a three-layer pipeline, automated tests, and a formal academic citation:

```
@software{histwardb2026,  
  author = {Manto, Vincenzo and Dal Cero, Riccardo},  
  title  = {HistWarDB: A Dynamic Wikidata-Based Historical War Dataset},  
  year   = {2026},  
  url    = {https://github.com/VincenzoManto/histwardb}  
}
```

If there's a moral here, it's the usual one: a friend's "simple" request is often the best way to discover that a seemingly solved problem (surely there's a dataset for everything?) actually isn't, and that building the right tool — clean, tested, reproducible — ends up being worth more than the single CSV that was asked for in the first place.