The Journal of Open Source Software

ChemInformant: A Robust and Workflow-Centric Python Client for High-Throughput PubChem Access

Zhiang He [1]

1 Independent Researcher

DOI: 10.21105/joss.08341

Software

- Review
- Repository
- Archive

Editor: Evan Spotte-Smith

Reviewers:

- @cthoyt
- @LIVazquezS

Submitted: 14 April 2025
Published: 13 August 2025

Summary

ChemInformant is a Python client for high-throughput, programmatic access to PubChem. It streamlines automated data retrieval by converting large, mixed-type lists of chemical identifiers directly into analysis-ready Pandas DataFrames (The pandas development team, 2025). To ensure resilience, the package integrates persistent HTTP caching, automatic rate-limiting with exponential backoff retries, and runtime data validation using Pydantic (Colvin & others, 2025). By addressing critical limitations in existing tools, such as network instability and inefficient batch processing, ChemInformant offers up to a 48-fold performance increase in batch retrieval compared to the widely-used PubChemPy library, providing a more reliable and efficient component for the modern Python cheminformatics ecosystem.

Statement of Need

Automated cheminformatics workflows require robust and efficient data access, but researchers face recurring challenges with existing PubChem clients. Network reliability is a primary concern. The PubChem API (Kim et al., 2018) enforces dynamic rate limits, which can halt automated scripts (National Center for Biotechnology Information, 2024). Many clients, like PubChemPy (Swain, 2017), lack built-in request throttling, retries, or persistent caching, forcing users to implement boilerplate code to handle network errors and redundant requests.

Batch processing is also often inefficient. Workflows with mixed-type identifiers (e.g., names and CIDs) require manual pre-processing. Furthermore, a single invalid identifier in a large batch can cause an entire query to fail without clear error reporting, hindering data acquisition pipelines.

ChemInformant addresses these gaps by natively integrating these critical features. Its architecture provides built-in resilience and a workflow-centric design, allowing researchers to focus on analysis rather than the low-level mechanics of data retrieval.

State of the Field and Comparison

To contextualize `ChemInformant`, its features were compared against related tools including PubChemPy (Swain, 2017), PubChemR (Korkmaz et al., 2025), webchem (Szöcs et al., 2020), ChemSpiPy (Swain, 2018), and PubChem4J (Southern & Griffin, 2011) (**Table 1**). The maintenance status of some libraries is noteworthy; for instance, PubChemPy has not had a formal release since 2017.

Table 1: Comparative analysis of key features in mainstream chemical information clients.

Key Feature	**ChemInformant**	PubChemPy	PubChemR	webchem	ChemSpiPy	PubChem4J
Platform	**Python**	Python	R	R	Python	Java

Key Feature	**ChemInformant**	PubChemPy	PubChemR	webchem	ChemSpiPy	PubChem4J
Primary Database	**PubChem**	PubChem	PubChem	Multi-DB	ChemSpider	PubChem
Persistent Caching[1]	**Yes**	No	No	No	No	N/A
Rate-Limiting & Retries[2]	**Yes**	No	No	Partial	No	N/A
Batch Retrieval	**Yes**	Partial	Partial	Partial	Partial	Yes
Mixed Identifier Support	**Yes**	No	No	No	No	N/A
Fault Tolerance[3]	**Yes**	No	No	No	No	N/A
Automatic Pagination	**Yes**	Partial	No	No	No	N/A
Runtime Type Safety	**Yes**	No	Partial	No	No	Yes
Project Activity	**Active**	Inactive	Active	Active	Inactive	Archived

Notes: [1] **Persistent Caching**: Stores results locally to accelerate repeated queries. [2] **Rate-Limiting & Retries**: Manages API request limits and server errors for robust automation. [3] **Fault Tolerance**: Reports status per-item in batch queries, avoiding complete failure on single errors.

Performance Evaluation

To quantify `ChemInformant`'s performance, a benchmark was performed to retrieve six properties for 285 drug names. `PubChemPy` was selected as a baseline. Since `PubChemPy`'s batch interface requires CIDs, all names were first resolved to CIDs. The performance of both libraries was then timed on processing this list.

Table 2 summarizes the performance data for the 280 successfully resolved compounds.

Table 2: Performance benchmark results.

Scenario	Time (s)	Speed-up (vs. PubChemPy)
PubChemPy (batch interface)	6.50	1× (Baseline)
ChemInformant — Cold Cache	**1.40**	**4.6×**
ChemInformant — Warm Cache	**0.135**	**48×**

The initial `ChemInformant` batch query completed in **1.4 seconds**, a **4.6-fold** speed increase over `PubChemPy`. A subsequent query from the cache finished in **135 milliseconds**, a 48-fold speed-up relative to the baseline. This sub-200ms response is suitable for interactive applications. The benchmark script is available in the project repository.

Example Usage

`ChemInformant` offers a layered API for both single lookups and batch processing.

Convenience API Example (single lookup):

```python
import ChemInformant as ci

# Retrieve a single property for one identifier
cas_number = ci.get_cas("ibuprofen")
# > '15687-27-1'
```

Core API Example (batch data analysis):

```python
# Retrieve multiple properties for a list of mixed-identifier types
df = ci.get_properties(
    identifiers=["aspirin", "caffeine", 1983], # Mix of names and a CID
    properties=["molecular_weight", "xlogp", "cas"]
)
# The returned DataFrame is formatted for direct use
print(df)
```

A detailed user manual is available in the project repository.

Acknowledgements

The author thanks PubChem for providing open data services (Kim et al., 2023) and acknowledges the developers of the open-source libraries upon which ChemInformant is built, including requests, pandas (The pandas development team, 2025), pydantic (Colvin & others, 2025), requests-cache (Haritonov & Cook, 2024), and pystow (Hoyt et al., 2025).

References

Colvin, S., & others. (2025). *Pydantic* (Version 2.11.7). Zenodo. https://doi.org/10.5281/zenodo.15662245

Haritonov, R., & Cook, J. W. (2024). *Requests-cache: Persistent HTTP cache for python requests* (Version 1.2.1). GitHub. https://github.com/requests-cache/requests-cache

Hoyt, C. T., Obraczka, D., Berrendorf, M., & Baird, S. G. (2025). *Cthoyt/pystow: v0.7.1* (Version v0.7.1). Zenodo. https://doi.org/10.5281/zenodo.15720682

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., & Bolton, E. E. (2023). PubChem 2023 update. *Nucleic Acids Research*, *51*(D1), D1373–D1380. https://doi.org/10.1093/nar/gkac956

Kim, S., Thiessen, P. A., Cheng, T., Yu, B., & Bolton, E. E. (2018). An update on PUG-REST: RESTful interface for programmatic access to PubChem. *Nucleic Acids Research*, *46*(W1), W563–W570. https://doi.org/10.1093/nar/gky294

Korkmaz, S., Yamasan, B. E., & Goksuluk, D. (2025). *PubChemR: An r client for the PubChem API* (Version 2.1.4). Comprehensive R Archive Network (CRAN). https://doi.org/10.32614/CRAN.package.PubChemR

National Center for Biotechnology Information. (2024). *PubChem programmatic access usage policy*. https://pubchem.ncbi.nlm.nih.gov/docs/programmatic-access

Southern, M. R., & Griffin, P. R. (2011). A java API for working with PubChem datasets. *Bioinformatics*, *27*(5), 741–742. https://doi.org/10.1093/bioinformatics/btq715

Swain, M. (2017). *PubChemPy v1.0.4* (Version 1.0.4). Zenodo. https://doi.org/10.5281/zenodo.541438

Swain, M. (2018). *ChemSpiPy: A python wrapper for the ChemSpider API* (Version 2.0.0). https://pypi.org/project/ChemSpiPy/2.0.0

Szöcs, E., Stirling, T., Scott, E. R., Scharmüller, A., & Schäfer, R. B. (2020). Webchem: An r package to retrieve chemical information from the web. *Journal of Statistical Software*, *93*(13), 1–17. https://doi.org/10.18637/jss.v093.i13

The pandas development team. (2025). *Pandas-dev/pandas: pandas* (Version 2.3.1). Zenodo. https://doi.org/10.5281/zenodo.15831829