The Journal of Open Source Software

taurenmd: A command-line interface for analysis of Molecular Dynamics simulations.

João M.C. Teixeira[1, 2]

1 Previous, Biomolecular NMR Laboratory, Organic Chemistry Section, Inorganic and Organic Chemistry Department, University of Barcelona, Baldiri Reixac 10-12, Barcelona 08028, Spain **2** Current, Program in Molecular Medicine, Hospital for Sick Children, Toronto, Ontario M5G 0A4, Canada

DOI: 10.21105/joss.02175

Software

- Review
- Repository
- Archive

Editor: Richard Gowers

Reviewers:

- @amritagos
- @luthaf

Submitted: 03 March 2020
Published: 01 June 2020

Summary

Molecular dynamics (MD) simulations of biological molecules have evolved drastically since its application was first demonstrated four decades ago (McCammon, Gelin, & Karplus, 1977) and, nowadays, simulation of systems comprising millions of atoms is possible due to the latest advances in computation and data storage capacity – and the scientific community's interest is growing (Hospital, Battistini, Soliva, Gelpí, & Orozco, 2019). Academic groups develop most of the MD methods and software for MD data handling and analysis. The MD analysis libraries developed solely for the latter scope nicely address the needs of manipulating raw data and calculating structural parameters, such as: `MDAnalysis` (Gowers et al., 2016; Michaud-Agrawal, Denning, Woolf, & Beckstein, 2011); `MDTraj` (McGibbon et al., 2015); `LOOS` (Romo, Leioatts, & Grossfield, 2014); and `PyTraj` (Hai Nguyen, 2016; Roe & Cheatham, 2013), each with its advantages and drawbacks inherent to their implementation strategies. This diversity enriches the field with a panoply of strategies that the community can utilize.

The MD analysis software libraries widely distributed and adopted by the community share two main characteristics: 1) they are written in pure Python (Rossum, 1995), or provide a Python interface; and 2) they are *libraries*: highly versatile and powerful pieces of software that, however, require advanced scripting and understanding to be operated, even for their basic functionalities. While this is the correct approach to develop flexible computational libraries, it creates a barrier between these software packages and *non-developer* researchers and hinder high throughput practices, particularly for routine data handling. Therefore, the need has emerged to create a platform that can efficiently combine the MD libraries available in the Python universe, taking the most out of each, and implements rapid interfaces for routine usage by both experts and non-experts in the field. In response to that, here is presented **taurenmd** (Figure 1), an easy-to-use and extensible ecosystem of command-line interfaces that facilitates complex operations in Molecular Dynamics data analysis by building on top of powerful Python-based MD analysis libraries.

The MD community has software packages that provide graphical interfaces to manipulate trajectories iteratively, such as VMD (Humphrey, Dalke, & Schulten, 1996), and the new integration between MDAnalysis and PyMol (Paul Smith, 2019). Also, there are powerful libraries to prepare molecular systems and run MD simulations, such as GROMACS (Abraham et al., 2015) and PLUMED (Bonomi et al., 2009). To date, *taurenmd* does not operates in either of these realms. If similar functionalities are developed in future versions, *taurenmd* will make use of the library interfaces provided by the projects referred before and will do so by creating command-line accessible pre-configured work-flows. Projects like GROMACS (Abraham et al., 2015) and LOOS (Romo et al., 2014) implement command-line ready interfaces to operate upon trajectories, differently to those libraries, *taurenmd* does not implement

core functionalities and, instead, it leverages the features provided by the referred third-party libraries. Also, *taurenmd* is a Python written software and, therefore, focuses on the usage of MD libraries that are also written in Python or provide a Python interface.

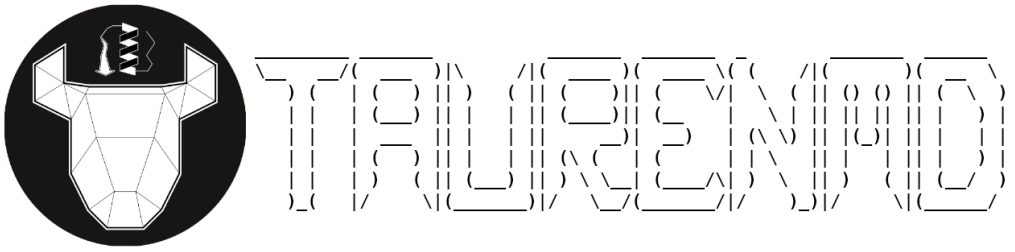

Figure 1: taurenmd logo.

Implementation

taurenmd provides highly parametrizable command-line interfaces that automate complex operations of Molecular Dynamics (MD) data handling and analysis in unitary executions that represent conceptual ideas, such as the manipulation of raw MD data or the calculation of structural parameters (*RMSDs, RMSFs, etc...*). Command-line operations are workflows defined by orchestrated single-operation functions. These single-logic functions are coded in the core of *taurenmd*'s library which facilitates unit-testing and sharing among all interfaces. Therefore, the design of *taurenmd*'s architecture is highly modular; yet, simple, flat, easy to read, and extensible. *taurenmd* serves as a continuous growing hub of routines, where new operations can be implemented and shared among the community in a defined and documented manner. The *taurenmd* project is hosted at GitHub (https://github.com/joaomcteixeira/taurenmd) and is extensively documented at ReadTheDocs (https://taurenmd.readthedocs.io).

To operate on MD data, *taurenmd* uses third-party MD analysis libraries; currently, it imports MDAnalysis (Gowers et al., 2016; Michaud-Agrawal et al., 2011), MDTraj (McGibbon et al., 2015), and OpenMM (Eastman et al., 2017), and they are used depending on the requirements of each command-line client. But, the design of the program allows facile incorporation of new dependencies to extend or implement new workflows. Finally, though *taurenmd* focuses on enhanced combination of third party libraries, its design leaves room for the implementation of original analysis routines when needed.

The command-line interface of *taurenmd* is hierarchic, where `taurenmd` is the main entry point and the different interfaces exist as subroutines, for example:

```
# help instructions for the main taurenmd entry point
$ taurenmd -h
# an execution example
$ taurenmd [SUBROUTINE] [OPTIONS]
# querying help for a specific subroutine
$ taurenmd report -h
```

At the date of publication, *taurenmd* provides ten different command-line interfaces; all of which, with their arguments and functionalities, are thoroughly described in the project's documentation under the "Command-line interfaces" section. Similarly, all individual functional

operations provided are open, fully documented, and can be imported and used by other projects if desired.

To invite community contributions, a client template file is provided with detailed instructions to guide the implementation of new command-line workflows. The building blocks required to build command-line clients are also extensively documented in the `libs/libcli.py` module, they are reusable and new blocks can also be added if needed. New logical operations can be implemented in the library core and used in clients. Complete instructions on how to contribute to the project are provided in the documentation. The project provides extensive Continuous Integration tests and explicit instructions for code style and format to guide developers. *taurenmd* follows Semantic Versioning 2.0 and we favor agile development/deployment instead of periodic releases.

Installation

To install **taurenmd** follow the instructions provided in the corresponding documentation page. *taurenmd* code uses only Python-provided interfaces and is, therefore, compatible with any platform able to execute Python. However, the different Molecular Dynamics analysis libraries imported have very different deployment strategies and the *taurenmd* project cannot guarantee those will function in all operating systems; it is, however, guaranteed that *taurenmd* works fully on Ubuntu 18.04 LTS running Anaconda as Python package manager. We advise reading the detailed installation instructions provided in the project's documentation.

Use cases

The *taurenmd* current version has ten command-line interfaces that execute different analysis or data manipulation routines. Extensive usage examples are provided in the documentation website or by the command:

```
$ taurenmd -h
```

Here we show how the `trajedit` interface is used for data manipulation and transformation:

```
$ taurenmd trajedit topology.pdb trajectory.xtc -d traj_s50_e500_p10.xtc \
> -s 50 -e 500 -p 10 -l 'segid A'
```

The latter extracts a subtrajectory spanning frames 50 to 500 (exclusive) with a step interval of 10 frames, and only for atoms for the `'segid A'` atom group; in this particular case, we make use of MDAnalysis library (Gowers et al., 2016; Michaud-Agrawal et al., 2011) to handle the data. *taurenmd* documentation provides additional usage examples in "Usage" page. Also, each client (command) documentation presents explanations on, and examples of, how to use it - see "Command-line interfaces" documentation page.

Acknowledgements

The initial concept of this project was largely inspired in the pdb-tools project "one script one action" idea (Rodrigues, Teixeira, Trellet, & Bonvin, 2018). The author deeply thanks João P.G.L.M. Rodrigues (ORCID: 0000-0001-9796-3193) for mentoring regarding MD simulations and data analysis and to Susana Barrera-Vilarmau (ORCID: 0000-0003-4868-6593) for her

intensive usage of the program since the very first versions and all the discussions, feedback and suggestions on building a user-friendly interface. The project's repository layout and Continuous Integration setup was based on `cookiecutter-pylibrary` repository (Mărieș, 2019) with final personal modifications by J.M.C.T.

References

Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, *1-2*, 19–25. doi:10.1016/j.softx.2015.06.001

Bonomi, M., Branduardi, D., Bussi, G., Camilloni, C., Provasi, D., Raiteri, P., Donadio, D., et al. (2009). PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Computer Physics Communications*, *180*(10), 1961–1972. doi:10.1016/j.cpc.2009.05.011

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L. P., et al. (2017). OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, *13*(7), 1–17. doi:10.1371/journal.pcbi.1005659

Gowers, Linke, Barnoud, Reddy, Melo, Seyler, Domański, et al. (2016). MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In Sebastian Benthall & Scott Rostrup (Eds.), *Proceedings of the 15th Python in Science Conference* (pp. 98–105). doi:10.25080/Majora-629e541a-00e

Hai Nguyen, J. S., Daniel R. Roe. (2016). pytraj. Retrieved from https://github.com/Amber-MD/pytraj

Hospital, A., Battistini, F., Soliva, R., Gelpí, J. L., & Orozco, M. (2019). Surviving the deluge of biosimulation data. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, (July 2019), 1–20. doi:10.1002/wcms.1449

Humphrey, W., Dalke, A., & Schulten, K. (1996). VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, *14*, 33–38.

Mărieș, I. C. (2019). cookiecutter-pylibrary. Retrieved from https://github.com/ionelmc/cookiecutter-pylibrary

McCammon, J. A., Gelin, B. R., & Karplus, M. (1977). Dynamics of folded proteins. *Nature*, *267*(5612), 585–590. doi:10.1038/267585a0

McGibbon, R. T., Beauchamp, K. A., Harrigan, M. P., Klein, C., Swails, J. M., Hernández, C. X., Schwantes, C. R., et al. (2015). MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical journal*, *109*(8), 1528–32. doi:10.1016/j.bpj.2015.08.015

Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: a toolkit for the analysis of molecular dynamics simulations. *Journal of computational chemistry*, *32*(10), 2319–27. doi:10.1002/jcc.21787

Paul Smith, M. B. (2019). MD Trajectories in PyMOL: No Memory Limits. Retrieved from https://nms.kcl.ac.uk/lorenz.lab/wp/?p=1768

Rodrigues, J. P. G. L. M., Teixeira, J. M. C., Trellet, M., & Bonvin, A. M. J. J. (2018). Pdb-tools: A swiss army knife for molecular structures. *F1000Research*, *7*, 1961. doi:10.12688/f1000research.17456.1

Roe, D. R., & Cheatham, T. E. (2013). PTRAJ and CPPTRAJ: Software for processing and analysis of molecular dynamics trajectory data. *Journal of Chemical Theory and Computation*, *9*(7), 3084–3095. doi:10.1021/ct400341p

Romo, T. D., Leioatts, N., & Grossfield, A. (2014). Lightweight object oriented structure analysis: tools for building tools to analyze molecular dynamics simulations. *Journal of computational chemistry*, *35*(32), 2305–2318. doi:10.1002/jcc.23753

Rossum, G. van. (1995). *Python tutorial* (No. CS-R9526). Amsterdam: Centrum voor Wiskunde en Informatica (CWI).