MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations

Richard J. Gowers[‖**†], Max Linke[‡‡†], Jonathan Barnoud[§†], Tyler J. E. Reddy[‡], Manuel N. Melo[§], Sean L. Seyler[¶], Jan Domanski[‡], David L. Dotson[¶], Sébastien Buchoux[††], Ian M. Kenney[¶], Oliver Beckstein[¶*]

Abstract—MDAnalysis (http://mdanalysis.org) is an library for structural and temporal analysis of molecular dynamics (MD) simulation trajectories and individual protein structures. MD simulations of biological molecules have become an important tool to elucidate the relationship between molecular structure and physiological function. Simulations are performed with highly optimized software packages on HPC resources but most codes generate output trajectories in their own formats so that the development of new trajectory analysis algorithms is confined to specific user communities and widespread adoption and further development is delayed. The MDAnalysis library addresses this problem by abstracting access to the raw simulation data and presenting a uniform object-oriented Python interface to the user. It thus enables users to rapidly write code that is portable and immediately usable in virtually all biomolecular simulation communities. The user interface and modular design work equally well in complex scripted workflows, as foundations for other packages, and for interactive and rapid prototyping work in IPython [PG07] / Jupyter notebooks, especially together with molecular visualization provided by nglview and time series analysis with pandas [McK10]. MDAnalysis is written in Python and Cython and uses NumPy [VCV11] arrays for easy interoperability with the wider scientific Python ecosystem. It is widely used and forms the foundation for more specialized biomolecular simulation tools. MDAnalysis is available under the GNU General Public License v2.

Index Terms—molecular dynamics simulations, science, chemistry, physics, biology

Introduction

Molecular dynamics (MD) simulations of biological molecules have become an important tool to elucidate the relationship between molecular structure and physiological function. Simulations are performed with highly optimized software packages on HPC resources but most codes generate output trajectories in their own formats so that the development of new trajectory analysis algorithms is confined to specific user communities and widespread adoption and further development is delayed. Typical trajectory sizes range from gigabytes to terabytes so it is typically not feasible to convert trajectories into a range of different formats just to use a tool that requires this specific form. Instead, a framework is required that provides a common interface to raw simulation data.

Results

The MDAnalysis library [MADWB11] addresses this problem by abstracting access to the raw simulation data and presenting a uniform object-oriented Python interface to the user. MDAnalysis is written in Python and Cython and uses NumPy arrays for easy interoperability with the wider scientific Python ecosystem. It currently supports more than 25 different file formats and covers the vast majority of data formats that are used in the biomolecular simulation community, including the formats required and produced by the most popular packages NAMD, Amber, Gromacs, CHARMM, LAMMPS, DL_POLY, HOOMD. The user interface provides "physics-based" abstractions (e.g. "atoms", "bonds", "molecules") of the data that can be easily manipulated by the user. It hides the complexity of accessing data and frees the user from having to implement the details of different trajectory and topology file formats (which by themselves are often only poorly documented and just adhere to certain "community expectations" that can be difficult to understand for outsiders).

The user interface and modular design work equally well in complex scripted workflows, as foundations for other packages like ENCORE [TPB+15] and ProtoMD *Somogyi:2016aa*, and for interactive and rapid prototyping work in IPython/Jupyter notebooks, especially together with molecular visualization provided by nglview and time series analysis with pandas. Since the original publication [MADWB11], improvements in speed and data structures make it now possible to work with terabyte-sized trajectories containing up to ~10 million particles. MDAnalysis also comes with specialized analysis classes in the MDAnalysis.analysis module that are unique to MDAnalysis such as the LeafletFinder graph-based algorithm for the analysis of lipid bilayers [MADWB11] or the Path Similarity Analysis for the quantitative comparison of macromolecular conformational changes [SKTB15].

MDAnalysis is available in source form under the GNU General Public License v2 from GitHub https://github.com/MDAnalysis/mdanalysis, PyPi and as conda packages. The documentation is extensive http://docs.mdanalysis.org including an introductory tutorial http://www.mdanalysis.org/MDAnalysisTutorial/. Our develoment community is very active with over 5 active core developers and lots of community contributions every release. We use modern software development

† *These authors contributed equally.*
‖ *University of Manchester, Manchester, UK*
** *University of Edinburgh, Edinburgh, UK*
‡‡ *Max Planck Institut für Biophysik, Frankfurt, Germany*
§ *University of Groningen, Groningen, The Netherlands*
‡ *University of Oxford, Oxford, UK*
¶ *Arizona State University, Tempe, Arizona, USA*
†† *Université de Picardie Jules Verne, Amiens, France*
* *Corresponding author: oliver.beckstein@asu.edu*

practices with continous integration and an extensive testsuite, >3500 tests and >92% for our core modules. If you like to use MDAnalysis for your project please join our community board.

Analysis Module

In the MDAnalysis.analysis module we provide a large variety of standard analysis algorithms, like RMSD, alignment [LAT10], native contacts [BHE13], [FKDD07], as well as unique algorithms, like the LeaftleftFinder [MADWB11] and Path Similarity Analysis [SKTB15]. Historically these algorithms were contributed by various researchers as individual moduldes to satisfy their own needs but this lead to some fragmentation in the user interface of these modules. We have recently started to unify the interface to the different algorithms with an *AnalysisBase* class. Currently PersistenceLength, InterRDF, LinearDensity and Contacts analysis have been ported. PersistenceLength calculates the persistence length of a polymer, InterRDF calculates the pairwise radial distribution function inside of a molecule, LinearDensity generates a density along a given exis and Contacts analysis native contacts, as desribed in more detail below. If applicable we also strive to make the API's to the algorithms generic. Most other tools hand the user analysis algorithms as black boxes. We want to avoid that and give the users all he needs to adapt an analysis to his/her needs.

The new Contacts class is a good example a generic API that allows easy adaptations of algorithms while still offering an easy setup for standard analysis types. The Contacts class is calculating a contact map for atoms in a frame and compares it with a reference map using different metrics. The used metric then decides which quantity is measued. A common quantity of interest is the fraction of native contacts, native contacts are all atoms that are nearby in the reference. For native contacts there exists two metrics [BHE13], [FKDD07] and we default to the later. We have designed the API to choose between the two metrics and pass user defined functions to develop new metrics or measure other quantities. This generic interface allowed us to implement a q1q2 analysis [FKDD07] on top of the Contacts class. Below is incomplete code example that shows how to implement a q1q2 analysis, the default value for the *method* kwarg is overwriten with a user defined method *radius_cut_q*. A more detailed explanatain can be found in the docs.

```
def radius_cut_q(r, r0, radius):
    y = r <= radius
    return y.sum() / r.size

contacts = Contacts(u, selection,
                    (first_frame_refs, last_frame_refs),
                    radius=radius, method=radius_cut_q,
                    start=start, stop=stop, step=step,
                    kwargs={'radius': radius})
```

This type of flexible analysis algorithms paired with a collection of base classes allow quick and easy analysis of simulations as well as development of new algorithms.

Conclusions

MDAnalysis provides a uniform interface to simulation data, which comes in a bewildering array of formats. It enables users to rapidly write code that is portable and immediately usable in virtually all biomolecular simulation communities. It has a very active international developer community with researchers that are expert developers and users of a wide range of simulation codes. MDAnalysis is widely used (the original paper [MADWB11] has been cited more than 180 times) and forms the foundation for more specialized biomolecular simulation tools. Ongoing and future developments will improve performance further, introduce transparent parallelisation schemes to utilize multi-core systems efficiently, and interface with the SPIDAL library for high performance data analytics algorithms.

References

[BHE13] Robert B Best, Gerhard Hummer, and William A Eaton. Native contacts determine protein folding mechanisms in atomistic simulations. *Proc. Natl. Acad. Sci. U. S. A.*, 110(44):17874–9, 2013. URL: http://www.pnas.org/content/110/44/17874, doi:10.1073/pnas.1311599110.

[FKDD07] Joel Franklin, Patrice Koehl, Sebastian Doniach, and Marc Delarue. MinActionPath: Maximum likelihood trajectory for large-scale structural transitions in a coarse-grained locally harmonic energy landscape. *Nucleic Acids Res.*, 35(SUPPL.2):477–482, 2007. doi:10.1093/nar/gkm342.

[LAT10] Pu Liu, Dimitris K Agrafiotis, and Douglas L. Theobald. Fast Determination of the Optimal Rotational Matrix for Macromolecular Superpositions. *J. Comput. Chem.*, 31(7):1561–1563, 2010. arXiv:NIHMS150003, doi:10.1002/jcc.

[MADWB11] Naveen Michaud-Agrawal, Elizabeth Jane Denning, Thomas B. Woolf, and Oliver Beckstein. MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *J Comp Chem*, 32:2319–2327, 2011. doi:10.1002/jcc.21787.

[McK10] Wes McKinney. Data Structures for Statistical Computing in Python. *Proc. 9th Python Sci. Conf.*, 1697900(Scipy):51–56, 2010. URL: http://conference.scipy.org/proceedings/scipy2010/mckinney.html.

[PG07] Fernando Pérez and Brian E. Granger. IPython: A system for interactive scientific computing. *Comput. Sci. Eng.*, 9(3):21–29, 2007. doi:10.1109/MCSE.2007.53.

[SKTB15] Sean L. Seyler, Avishek Kumar, M. F. Thorpe, and Oliver Beckstein. Path similarity analysis: A method for quantifying macromolecular pathways. *PLoS Comput Biol*, 11(10):e1004568, 10 2015. URL: http://dx.doi.org/10.1371%2Fjournal.pcbi.1004568, doi:10.1371/journal.pcbi.1004568.

[TPB+15] Matteo Tiberti, Elena Papaleo, Tone Bengtsen, Wouter Boomsma, and Kresten Lindorff-Larsen. ENCORE: Software for quantitative ensemble comparison. *PLoS Comput Biol*, 11(10):e1004415, 10 2015. URL: http://dx.doi.org/10.1371%2Fjournal.pcbi.1004415, doi:10.1371/journal.pcbi.1004415.

[VCV11] Stefan Van Der Walt, S. Chris Colbert, and Gael Varoquaux. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, 13(2):22–30, 2011. arXiv:1102.1523, doi:10.1109/MCSE.2011.37.