Computational Tools

MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories

Robert T. McGibbon,[1,*] Kyle A. Beauchamp,[2] Matthew P. Harrigan,[1] Christoph Klein,[3] Jason M. Swails,[4] Carlos X. Hernández,[5] Christian R. Schwantes,[1] Lee-Ping Wang,[6] Thomas J. Lane,[7] and Vijay S. Pande[1,5]
[1]Department of Chemistry, Stanford University, Stanford, California; [2]Computational Biology Program, Sloan-Kettering Institute, New York, New York; [3]Department of Chemical and Biomolecular Engineering, Vanderbilt University, Nashville, Tennessee; [4]Department of Chemistry, Rutgers University, Piscataway, New Jersey; [5]Biophysics Program, Stanford University, Stanford, California; [6]Department of Chemistry, University of California, Davis, Davis, California; and [7]SLAC National Accelerator Laboratory, Menlo Park, California

ABSTRACT As molecular dynamics (MD) simulations continue to evolve into powerful computational tools for studying complex biomolecular systems, the necessity of flexible and easy-to-use software tools for the analysis of these simulations is growing. We have developed MDTraj, a modern, lightweight, and fast software package for analyzing MD simulations. MDTraj reads and writes trajectory data in a wide variety of commonly used formats. It provides a large number of trajectory analysis capabilities including minimal root-mean-square-deviation calculations, secondary structure assignment, and the extraction of common order parameters. The package has a strong focus on interoperability with the wider scientific Python ecosystem, bridging the gap between MD data and the rapidly growing collection of industry-standard statistical analysis and visualization tools in Python. MDTraj is a powerful and user-friendly software package that simplifies the analysis of MD data and connects these datasets with the modern interactive data science software ecosystem in Python.

INTRODUCTION

Molecular dynamics (MD) simulations yield a great deal of information about the structure, dynamics, and function of biological macromolecules by modeling the physical interactions between their atomic constituents. Modern MD simulations, often using distributed computing, graphics processing unit acceleration, or specialized hardware can generate large datasets containing hundreds of gigabytes or more of trajectory data tracking the positions of a system's atoms over time (1). To use these vast and information-rich datasets to understand biomolecular systems and generate scientific insight, further computation, analysis, and visualization are required (2).

Within the last decade, the Python language (https://www.python.org/) has become a major hub for scientific computing. It features a wealth of high-quality open source packages, including those for interactive computing (3), machine learning (4), and visualization (5). This environment is ideal for both rapid development and high performance, as computational kernels can be implemented in the languages C, C++, and FORTRAN, but made available within a more user-friendly interactive environment.

In the MD community, the benefits of integration with such industry standard tools have not yet been fully realized because of a tradition of custom file formats and command-line analysis. To address this need, we have developed MDTraj, a modern, open, and lightweight Python library for analysis and manipulation of MD trajectories. The project has the following goals:

1) To serve as a bridge between MD data and the modern statistical analysis and scientific visualization software ecosystem in Python.
2) To support a wide range of MD data formats and computations.
3) To run rapidly on modern hardware with efficient memory utilization, enabling the interactive analysis of large datasets.

Several other software packages for the analysis of MD trajectories exist, including the GROMACS tools (6), CPPTRAJ (7), VMD (8), MMTK (9), MDAnalysis (10), Bio3D (11), ST-Analyzer (12), LOOS (13), and Pteros (14). GROMACS and CPPTRAJ provide a broad range of functionality to users from the Unix command line, or with a simple interactive scripting environment. LOOS and Pteros are C++ toolkits that enable the construction of novel trajectory analysis programs, while VMD and ST-Analyzer provide convenient graphical interfaces. Like MDTraj, MMTK and MDAnalysis are written in Python while Bio3D is written in the statistical programming language R (https://www.r-project.org/). Each of these software packages has capabilities that have served to inform the development of MDTraj.

Submitted June 23, 2015, and accepted for publication August 10, 2015.
*Correspondence: rmcgibbo@stanford.edu
Editor: David Sept.

0006-3495/15/10/1528/5
http://dx.doi.org/10.1016/j.bpj.2015.08.015

MATERIALS AND METHODS

Capabilities and implementation

MDTraj is widely interoperable and extremely easy to use. First and foremost, MDTraj can load trajectory and/or topology data from the formats used by a broad range of MD packages, including AMBER (15), GROMACS (6), DESMOND (16), CHARMM (17), NAMD (18), TINKER (19), LAMMPS (20), OpenMM (21), ACEMD (22), and HOOMD-Blue (23); see Table 1 for a full list of supported file formats. This wide support enables consistent interfaces and reproducible analyses regardless of users' preferred MD simulation packages.

From its inception, MDTraj has been designed to work in concert with other packages for analysis and visualization. No single toolkit can provide all possible ways to analyze molecular simulations, especially given the rapid pace of development in statistics and data science. Rather than attempting to provide all conceivable functionality in one toolkit, MDTraj leverages Python and NumPy (http://www.numpy.org/) to empower users to connect their MD data with the large and rapidly growing ecosystem of data science tools available more broadly in the community.

MDTraj originated from the trajectory handling portions of MSMBuilder (24), where it now provides a stable base for handling trajectories, computing order parameters and projections, and providing the distance metrics—such as minimal root-mean-squared deviation (RMSD)—that are necessary for clustering. Additionally, it is now used inside tools that analyze data from the Folding@home distributed computing architecture (25), a structure-based virtual screening pipeline at Google Research, the PyEMMA Markov modeling package (26), the Ensembler and mBuild (27,28) modeling tools, and countless individual analysis scripts. MDTraj is part of the Omnia consortium (http://omnia.md) suite of tools, which will be described in a later article.

Most data analyses for MD involve either extracting a vector of order parameters of each simulation snapshot or defining a distance metric between snapshots. MDTraj makes it very easy to rapidly extract these representations. It includes an extremely fast RMSD engine capable of operating near the machine floating point limit described in detail by Haque et al. (29), performing Theobald's QCP algorithm (30) approximately three times faster than the original implementation. Functions for secondary-structure assignment (31), solvent-accessible surface area determination (32), hydrogen bond identification (33), residue-residue contact mapping, NMR scalar coupling constants (34), nematic order parameters (35), and the extraction of various internal degrees of freedom are similarly available. Where appropriate, these compute kernels are written in C or C++ and heavily optimized with vectorized instructions and multithreading. To enable interoperability, these data are returned to the user as multidimensional NumPy arrays, the standard numeric data storage format for the scientific Python ecosystem.

MDTraj also provides an atom selection language. Often, analysis functions are applied to a subset of atoms in the system. To generate arrays of these indices, the topology attribute and full Python grammar can be a powerful combination (i.e., Fig. 1, line 2). For users less familiar with Python or making the transition from other packages, a natural text-based selection syntax can be used as well (i.e., Fig. 1, line 3). These selection strings can be translated into standard Python syntax for pedagogical purposes or directly executed.

Ease-of-use is a central and deliberate goal at each level of the design and implementation of MDTraj. This starts with installation. Using the cross-platform Conda package manager, users can get started in seconds using the shell command `conda install -c omnia mdtraj`, which downloads and installs precompiled binaries of MDTraj (and all of its dependencies) on Windows, Linux, or Mac OS-X, without the requirement of administrator privileges.

The package has an extremely simple object model, which makes it very easy for new users to get started. Only a single class, `Trajectory`, needs to be mastered; it contains all relevant information about the MD trajectory, such as the atomic coordinates, unit cell dimensions, and simulation time. Loading files and performing analysis are generally done with functions (e.g., `mdtraj.load`, `mdtraj.compute_<name>`) as opposed to classes to provide a simple and intuitive user experience that minimizes the need to remember complex object workflows.

MDTraj is extensively documented in a consistent format. The package itself contains over 9000 lines of Python docstrings that describe each function and class. The website, http://mdtraj.org, contains complete documentation, but more importantly contains 14 complete, executable example notebooks demonstrating topics including hydrogen-bond identification, Ramachandran plotting, and strategies for memory-limited computation on large datasets. These examples provide new users the patterns to get up and running with their own analyses immediately.

Furthermore, MDTraj includes a unique interactive WebGL-based three-dimensional structure viewer for the IPython notebook adapted from iview (36), shown in Fig. 2. Because it combines the analysis input code with results and plots into a single worksheet, the IPython notebook provides one of the most convenient user interfaces for interactive analysis. This convenience is further enhanced by MDTraj's `TrajectoryView` widget, which runs inside the IPython notebook and provides a high-quality and fully interactive three-dimensional rendering of a trajectory. The viewer can save high-quality `png` images or `STL` three-dimensional models. MDTraj thus not only provides first-class scriptability but also high-quality three-dimensional visualization.

The development, engineering, and testing of MDTraj incorporates modern best practices for scientific computing (37). The package contains more than 1100 unit tests for individual components. These tests are continually run on each incremental contribution on both Windows and Linux, using multiple versions of Python and the required libraries. The project is hosted on GitHub, and development takes place fully openly and collaboratively. Users of MDTraj are often researchers who are interested in analyzing simulations in new ways, a task that involves not only MDTraj library functions but also writing new code. The simple coding style, open source licensing, GitHub pull-request-based development pattern (38), and active culture of collaborative code review enable these researchers to rapidly prototype new methods and extend MDTraj. This has been borne out by the MDTraj community, which comprises members from numerous academic and industrial research groups across the world that have contributed to the project over the past two years.

TABLE 1 List of supported file formats

Package	File Formats
Many packages	`pdb, xyz, dcd`
Amber	`prmtop, crd, netcdf, binpos, restrt`
Gromacs	`gro, xtc, trr`
Desmond	`dtr, stk`
CHARMM	`psf`
LAMMPS	`lammpstrj`
TINKER	`arc`
HOOMD-Blue	`xml`
OpenMM	`xml`
TRIPOS	`mol2`
MDTraj	`hdf5`

RESULTS AND DISCUSSION

The capabilities of MDTraj serve as a bridge, connecting MD data with statistics and graphics libraries developed for general data science audiences. A key advantage of this design, for users and developers, is access to a much wider range of state-of-the-art analysis capabilities characterized by large feature sets, extensive documentation, and active user communities.

```
In [1]: top = my_traj.topology
In [2]: [atom.index for atom in top.atoms if atom.residue.is_water and atom.name == "O"]
In [3]: top.select("water and name O")
```

FIGURE 1 The MDTraj atom selection language. Queries can be expressed using standard Python code (line 2), or an intuitive string-based syntax (line 3).

A demonstration of this integrative workflow is shown in Fig. 3, which combines MDTraj with the `scikit-learn` (4) for principal component analysis (PCA) and `matplotlib` (5) for visualization, to determine high-variance collective motions in a protein system. While PCA is a widely used method that is included in a variety of MD analysis packages, the advantage of integrating with the wider data science community is immediately evident when moving on to more complex statistical analysis. For example, a variety of sparse and kernelized PCA-like methods have been introduced into the machine learning community (39), and may be quite powerful for analyzing more complex protein systems. Because of its open and interoperable design, these cutting-edge statistical tools are readily available to MD researchers with MDTraj, without duplication of developer efforts and independent of the particular MD software used to perform the simulations.

We generally find that file I/O and main memory are more limiting than raw CPU performance for MD analysis. For this reason, simple multinode parallelization, even over relatively slow interconnects, can often be extremely useful for accelerating calculations. As an example, Fig. 4 shows a demonstration of the use of MDTraj with the IPython parallel toolkit to parallelize the calculation of the solvent-accessible surface area of a trajectory over the individual snapshots of the trajectory. The code requires separately initializing an array of IPython engine processes on which the calculation is executed. These can be distributed over many nodes on a cluster or in the cloud and linked together by MPI or SSH. Because many simulation datasets contain many separate MD trajectories saved in separate files, a similar pattern can also be used to process individual files in parallel.

CONCLUSIONS

Within the field of trajectory analysis tools, MDTraj stands out due to its ease of use, flexibility, and Python-centric design, largely thanks to its organization around the intuitive `Trajectory` object in which data are stored as NumPy arrays. This design significantly enhances extensibility and gives users a great deal of latitude for freely accessing and manipulating the data according to the needs of their research. MDTraj speeds up analysis tasks by implementing computationally intensive operations (such as RMSD) using optimized low-level kernels written in C/C++. Furthermore, MDTraj can read and write a very wide range of trajectory file formats, ensuring interoperability across most MD software packages.

```
import mdtraj as md
from mdtraj.html import TrajectorySliderView

traj = md.load("trajectory.pdb")
TrajectorySliderView(traj, secondaryStructure="ribbon")
```

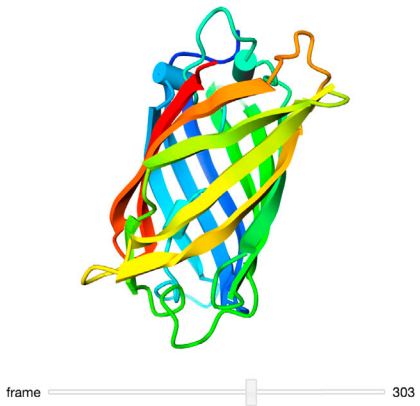

FIGURE 2 MDTraj's interactive WebGL-based protein and trajectory viewer. This feature requires a modern WebGL-enabled browser, and the IPython notebook that can be installed with Conda using the command `conda install IPython-notebook`. To see this figure in color, go online.

```
import mdtraj as md
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

t = md.load("trajectory.pdb")
pairs = t.top.select_pairs("all", "all")
X = md.compute_distances(t, pairs)

pca = PCA(n_components=2)
Y = pca.fit_transform(X)

plt.hexbin(Y[:, 0], Y[:, 1], bins="log")
```

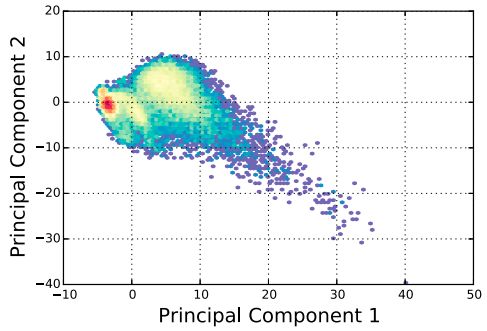

FIGURE 3 Demonstration of PCA with MDTraj, `scikit-learn`, and MATPLOTLIB. To see this figure in color, go online.

```
from IPython.parallel import Client
import numpy as np
import mdtraj as md
import matplotlib.pyplot as plt

c = Client()

traj = md.iterload("trajectory.pdb", chunk=10)
results = c[:].map(md.shrake_rupley, traj)
sasa = results.get()

plt.plot(np.sum(np.vstack(sasa), 1))
```

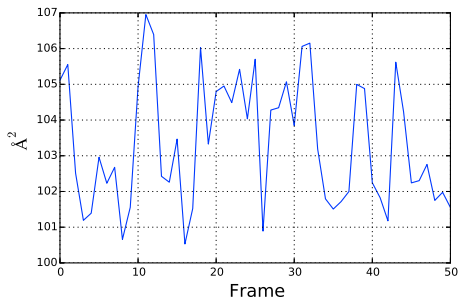

FIGURE 4 Demonstration of solvent-accessible surface area calculation done in parallel with MDTraj and IPython. To see this figure in color, go online.

Software Availability

MDTraj is available under the GNU Lesser General Public License (LGPL), version 2.1 or later. Full documentation and examples are available at the project home page, http://mdtraj.org, and development is hosted on GitHub at http://github.com/mdtraj/mdtraj. The latest release, version 1.4.2, is archived at doi:10.5281/zenodo.18700.

AUTHOR CONTRIBUTIONS

R.T.M., K.A.B., M.P.H., C.K., J.M.S., C.X.H., C.R.S., L.-P.W., and T.J.L. developed the software; R.T.M. drafted the article; R.T.M., C.X.H., M.P.H., L.-P.W., J.M.S., K.A.B., C.R.S., and T.J.L. edited the article; and all authors read and approved the final article.

ACKNOWLEDGMENTS

We are grateful to the full team of MDTraj contributors: Patrick Riley, Teng Lin, Tim Moore, Ravi Ramanathan, Joshua Adelman, Chaya Stern, Gert Kiss, Muneeb Sultan, Yutong Zhao, Andrea Zonca, Ondrej Marsalek, Thomas Peulen, Anton Goloborodko, and Alexander Götz, as well as participants on the MDTraj discussion forum and issue tracker.

The authors acknowledge funding from the National Institutes of Health (grants No. R01-GM62868 and No. P30-CA008748) and National Science Foundation (grant No. MCB-0954714).

REFERENCES

1. Klepeis, J. L., K. Lindorff-Larsen, …, D. E. Shaw. 2009. Long-time-scale molecular dynamics simulations of protein structure and function. *Curr. Opin. Struct. Biol.* 19:120–127.
2. Lane, T. J., D. Shukla, …, V. S. Pande. 2013. To milliseconds and beyond: challenges in the simulation of protein folding. *Curr. Opin. Struct. Biol.* 23:58–65.
3. Pérez, F., and B. E. Granger. 2007. IPython: a system for interactive scientific computing. *Comput. Sci. Eng.* 9:21–29.
4. Pedregosa, F., G. Varoquaux, …, E. Duchesnay. 2011. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12:2825–2830.
5. Hunter, J. D. 2007. Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* 9:90–95.
6. Hess, B., C. Kutzner, …, E. Lindahl. 2008. GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chem. Theory Comput.* 4:435–447.
7. Roe, D. R., and T. E. Cheatham. 2013. PTRAJ and CPPTRAJ: software for processing and analysis of molecular dynamics trajectory data. *J. Chem. Theory Comput.* 9:3084–3095.
8. Humphrey, W., A. Dalke, and K. Schulten. 1996. VMD: visual molecular dynamics. *J. Mol. Graph.* 14:33–38, 27–28.
9. Hinsen, K. 2000. The molecular modeling toolkit: a new approach to molecular simulations. *J. Comput. Chem.* 21:79–85.
10. Michaud-Agrawal, N., E. J. Denning, …, O. Beckstein. 2011. MDAnalysis: a toolkit for the analysis of molecular dynamics simulations. *J. Comput. Chem.* 32:2319–2327.
11. Grant, B. J., A. P. C. Rodrigues, …, L. S. D. Caves. 2006. Bio3D: an R package for the comparative analysis of protein structures. *Bioinformatics.* 22:2695–2696.
12. Jeong, J. C., S. Jo, …, W. Im. 2014. ST-Analyzer: a web-based user interface for simulation trajectory analysis. *J. Comput. Chem.* 35:957–963.
13. Romo, T., and A. Grossfield. 2009. LOOS: an extensible platform for the structural analysis of simulations. *In* Engineering in Medicine and Biology Society, EMBC 2009. Annual International Conference of the IEEE. Institute of Electrical and Electronics Engineers, Piscataway, NJ, pp. 2332–2335.
14. Yesylevskyy, S. O. 2012. Pteros: fast and easy to use open-source C++ library for molecular analysis. *J. Comput. Chem.* 33:1632–1636.
15. Case, D., T. Darden, …, P. Kollman. 2015. AMBER. University of California, San Francisco, CA.
16. Bowers, K., E. Chow, …, D. Shaw. 2006. Scalable algorithms for molecular dynamics simulations on commodity clusters. *In* ACM/IEEE SC 2006 Conference. Institute of Electrical and Electronics Engineers, New York, 43–43.
17. Brooks, B. R., R. E. Bruccoleri, …, M. Karplus. 1983. CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.* 4:187–217.
18. Phillips, J. C., R. Braun, …, K. Schulten. 2005. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* 26:1781–1802.
19. Ponder, J. W., and F. M. Richards. 1987. An efficient Newton-like method for molecular mechanics energy minimization of large molecules. *J. Comput. Chem.* 8:1016–1024.
20. Plimpton, S. 1995. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.* 117:1–19.
21. Eastman, P., M. S. Friedrichs, …, V. S. Pande. 2013. OpenMM 4: a reusable, extensible, hardware independent library for high performance molecular simulation. *J. Chem. Theory Comput.* 9:461–469.
22. Harvey, M. J., G. Giupponi, and G. D. Fabritiis. 2009. ACEMD: accelerating biomolecular dynamics in the microsecond time scale. *J. Chem. Theory Comput.* 5:1632–1639.
23. Anderson, J. A., C. D. Lorenz, and A. Travesset. 2008. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Comput. Phys.* 227:5342–5359.
24. Beauchamp, K. A., G. R. Bowman, …, V. S. Pande. 2011. MSMBuilder2: modeling conformational dynamics at the picosecond to millisecond scale. *J. Chem. Theory Comput.* 7:3412–3419.
25. Larson, S. M., C. D. Snow, …, V. S. Pande. 2004. Folding@home and Genome@home: using distributed computing to tackle previously intractable problems in computational biology. *In* Computational Genomics: Theory and Applications. R. Grant, editor. Horizon Scientific Press, Norfolk, VA.

26. Senne, M., B. Trendelkamp-Schroer, …, F. Noé. 2012. EMMA: a software package for Markov model building and analysis. *J. Chem. Theory Comput.* 8:2223–2238.

27. Parton, D. L., P. B. Grinaway, …, J. D. Chodera. 2015. Ensembler: enabling high-throughput molecular simulations at the superfamily scale. bioRxiv, 018036.

28. Klein, C. 2014. mBuild: a component-based molecule builder tool that relies on equivalence relations for component composition. *GitHub. http://imodels.github.io/mbuild/.*

29. Haque, I. S., K. A. Beauchamp, and V. S. Pande. 2014. A fast 3 × N matrix multiply routine for calculation of protein RMSD. bioRxiv, 008631.

30. Theobald, D. L. 2005. Rapid calculation of RMSDs using a quaternion-based characteristic polynomial. *Acta Crystallogr. A.* 61:478–480.

31. Kabsch, W., and C. Sander. 1983. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers.* 22:2577–2637.

32. Shrake, A., and J. A. Rupley. 1973. Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *J. Mol. Biol.* 79:351–371.

33. Baker, E. N., and R. E. Hubbard. 1984. Hydrogen bonding in globular proteins. *Prog. Biophys. Mol. Biol.* 44:97–179.

34. Vögeli, B., J. Ying, …, A. Bax. 2007. Limits on variations in protein backbone dynamics from precise measurements of scalar couplings. *J. Am. Chem. Soc.* 129:9377–9385.

35. Allen, M. P., and D. J. Tildesley. 1989. Liquid crystals. *In* Computer Simulation of Liquids. Clarendon Press, Oxford, UK, pp. 300–305.

36. Li, H., K.-S. Leung, …, M.-H. Wong. 2014. iview: an interactive WebGL visualizer for protein-ligand complex. *BMC Bioinformatics.* 15:56.

37. Wilson, G., D. A. Aruliah, …, P. Wilson. 2014. Best practices for scientific computing. *PLoS Biol.* 12:e1001745.

38. Gousios, G., M. Pinzger, and A. van Deursen. 2014. An exploratory study of the pull-based software development model. *In* Proceedings of the 36th International Conference on Software Engineering, ICSE 2014. Association for Computing Machinery (ACM), New York, pp. 345–355.

39. Burges, C. 2010. Dimension Reduction: A Guided Tour, Foundations and Trends in Machine Learning. Now Publishers, Boston, MA.