The Journal of Open Source Software

msmhelper: A Python package for Markov state modeling of protein dynamics

Daniel Nagel [1] **and Gerhard Stock** [1]

1 Biomolecular Dynamics, Institute of Physics, Albert-Ludwigs-Universität Freiburg, 79104 Freiburg, Germany

DOI: 10.21105/joss.05339

Software

- Review
- Repository
- Archive

Editor: Lucy Whalley

Reviewers:

- @lorenzo-rovigatti
- @taoliu032
- @yuxuanzhuang

Submitted: 16 March 2023
Published: 24 May 2023

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Proteins function through complex conformational changes that can be difficult to study experimentally. Molecular dynamics simulations provide high spatiotemporal resolution, but generate massive amounts of data that require specialized analysis. Markov state modeling is a common approach to interpret simulations, which involves identifying biologically relevant conformations and modeling the dynamics as memoryless transitions between them. Markov models can complement experimental data with atomistic information, leading to a deeper understanding of protein behavior.

In this work, we present `msmhelper`, a Python package that provides a user-friendly and computationally efficient implementation for estimating and validating Markov state models of protein dynamics. Given a set of metastable conformational states, the package offers a wide range of functionalities to improve model predictions, including dynamical correction techniques such as the Hummer-Szabo projection formalism, dynamical coring, and Gaussian filtering, as well as methods for predicting experimentally relevant timescales and pathways.

Statement of need

Markov state modeling (MSM) has emerged as an important tool for the analysis of molecular dynamics (MD) simulations of protein dynamics (Bowman et al., 2013; Buchete & Hummer, 2008; Prinz et al., 2011; Wang et al., 2018). In the general workflow of MSM, clustering into hundreds to thousands of microstates is crucial to accurately represent the free energy landscape and correct for non-optimal cuts between states. However, to comprehend the underlying biological processes, it is essential to cluster these microstates into a few macrostates that describe the dynamics as jumps between biologically relevant metastable conformations. Despite the assumption that microstate dynamics can be modeled by Markovian jumps, this is generally not the case for coarse-grained macrostate dynamics due to insufficient timescale separation between intrastate and interstate dynamics. To address this challenge, `msmhelper` includes various dynamical correction techniques, including a Gaussian filtering approach to include short-term time-information in the geometric-based clustering step (Nagel et al., 2023), dynamical coring (Nagel et al., 2019) to correct for spurious intrastate fluctuations at the barrier, misclassified as interstate fluctuations, and the Hummer-Szabo projection formalism (Hummer & Szabo, 2015), enabling an optimal projection of microstate dynamics onto the macrostate space. Moreover, `msmhelper` provides an easy-to-use interface for common MSM analyses, including the estimation of the transition matrix, the validation via Chapman-Kolmogorov tests and implied timescales, and the estimation of biological relevant pathways including their time scales, based on the concept of `MSMPathfinder` (Nagel et al., 2020).

There are well-established Python packages, in particular PyEMMA (Scherer et al., 2015) and MSMBuilder (Beauchamp et al., 2011), providing a comprehensive set of tools for the

entire MSM workflow, from feature extraction and projection of MD simulations onto collective variables, to clustering and dynamical lumping to identify metastable conformations, and ultimately to the estimation of Markov models and the prediction of relevant dynamics. In contrast to them, `msmhelper` focuses only on the estimation and analysis of Markov state models starting from given (micro- and or macro-) state trajectories. Furthermore—to the best of the authors' knowledge—this is the first publicly available Python implementation of some methodologies not implemented in `PyEmma` and `MSMBuilder`, including dynamical coring, the Hummer-Szabo projection formalism, and the estimation of waiting time based pathways. Additionally, this package provides a rich command-line interface for common analysis, including the creation of publication-ready figures of the implied timescales, the Chapman-Kolmogorov test, different visualizations of the waiting time distributions, and a comprehensive state representation. The latter two techniques were suggested in Nagel et al. (2023), which uses `msmhelper` to analyze protein dynamics.

Since Markov state modeling is usually done on local computers, it is important to provide sufficiently fast performance. By using `numpy` (Harris et al., 2020) and just-in-time compilation via `numba` (Lam et al., 2015), all performance-critical methods in `msmhelper` have been optimized. As a result, `msmhelper` can be much faster than conventional multi-purpose programs such as `PyEmma`. For example, adopting a 10-state trajectory with 10^5 time steps, both the run time of the MSM estimation (transition probability matrix) and its validation by the well-established Chapman-Kolmogorov test are more than an order of magnitude faster. If we compare the performance of the Markov chain Monte Carlo (MCMC) propagation, which is commonly used to determine pathways including their corresponding time scale distributions, `msmhelper` outperforms `PyEMMA` by up to two orders of magnitude. More details and additional benchmarks, including source code, can be found in the documentation.

Example

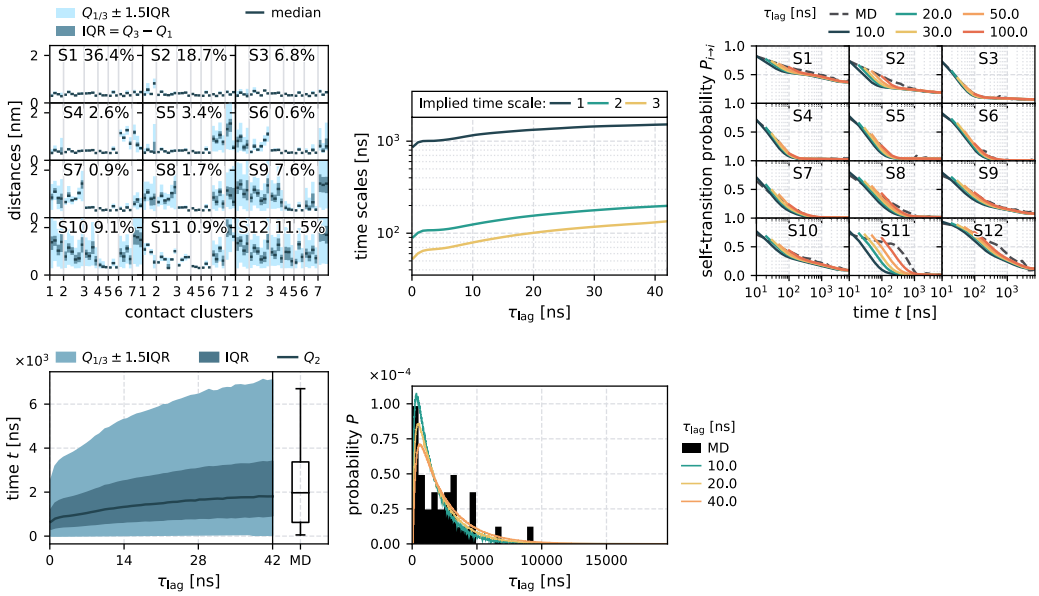

Figure 1: MSM of villin headpiece, data taken from Nagel et al. (2023). (top left) Compact structural representation of the states, called contact representation, (top center) implied timescales to validate the Markovianity of the model, (top right) Chapman-Kolmogorov test for models based on different lag times compared to the MD simulation, (bottom left) waiting time distribution of the folding time for varying lag times compared, and (bottom center) detailed comparison of folding time distributions to the MD simulation.

In the following, we briefly demonstrate the capabilities of the provided command-line interface. For this purpose, we use the publicly available micro- and macrostate trajectories of the villin headpiece (see, Nagel et al., 2023). It is a well-studied fast folding protein, that allows us to test common MSM analysis, including state characterization, MSM validation, and folding timescale estimation. All results shown in Figure 1 were generated directly from the command-line interface of `msmhelper`. (Top left) To characterize the structure of the metastable states, Nagel et al. (2023) introduced a contact-based representation, each state is described by the distribution of contacts within the contact clusters obtained using the correlation-based feature selection method MoSAIC (Diez et al., 2022). (Top center) When estimating a Markov state model, selecting a sufficiently long lag time τ_{lag} is of importance to ensure the Markovianity of the data (see, e.g., Prinz et al., 2011). That is, the implied timescales t_i should be constant, given by $t_i = -\tau_{\text{lag}} / \log(\lambda_i)$, where λ_i denotes the i-th largest left-handed eigenvalue. (Top right) Another approach to validate the Markovianity is the Chapman-Kolmogorov test which compares the model's predictions of time evolution with the results of molecular dynamics (MD) simulations. (Bottom) To relate to experiment, we compare the predicted folding time distributions with the MD simulation results. Using the median, interquartile range (IQR), and corresponding bounds.

Acknowledgements

The computationally efficient implementations of `msmhelper` are made possible by `numba` (Lam et al., 2015) and `numpy` (Harris et al., 2020), the visualizations are based on `matplotlib` (Hunter, 2007) and `prettypyplot` (Nagel, 2022), and the command-line interface is realized with `click` (The Pallets Projects, 2022).

Furthermore, the authors thank Georg Diez, Sofia Sartore, Miriam Jäger, Emanuel Dorbath, and Ahmed Ali for valuable discussions and testing the software package.

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) via the Research Unit FOR 5099 "Reducing complexity of nonequilibrium" (project No. 431945604). The authors acknowledge support by the High Performance and Cloud Computing Group at the Zentrum für Datenverarbeitung of the University of Tübingen and the Rechenzentrum of the University of Freiburg, the state of Baden-Württemberg through bwHPC and the DFG through Grant Nos. INST 37/935-1 FUGG (RV bw16I016) and INST 39/963-1 FUGG (RV bw18A004).

References

Beauchamp, K. A., Bowman, G. R., Lane, T. J., Maibaum, L., Haque, I. S., & Pande, V. S. (2011). MSMBuilder2: Modeling Conformational Dynamics on the Picosecond to Millisecond Scale. *J. Chem. Theory Comput.*, *7*(10), 3412–3419. https://doi.org/10.1021/ct200463m

Bowman, G. R., Pande, V. S., & Noé, F. (2013). *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*. Springer Netherlands. https://doi.org/10.1007/978-94-007-7606-7

Buchete, N.-V., & Hummer, G. (2008). Coarse master equations for peptide folding dynamics. *J. Phys. Chem. B*, *112*(19), 6057–6069. https://doi.org/10.1021/jp0761665

Diez, G., Nagel, D., & Stock, G. (2022). Correlation-based feature selection to identify functional dynamics in proteins. *J. Chem. Theory Comput.*, *18*(8), 5079–5088. https://doi.org/10.1021/acs.jctc.2c00337

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant,

T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hummer, G., & Szabo, A. (2015). Optimal Dimensionality Reduction of Multistate Kinetic and Markov-State Models. *J. Phys. Chem. B*, *119*(29), 9029–9037. https://doi.org/10.1021/jp508375q

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. https://doi.org/10.1145/2833157.2833162

Nagel, D. (2022). *Prettypyplot: Publication ready matplotlib figures made simple*. Zenodo. https://doi.org/10.5281/zenodo.7278311

Nagel, D., Satore, S., & Stock, G. (2023). *Selecting Features for Markov Modeling: A Case Study on HP35*. arXiv. https://doi.org/10.48550/arXiv.2303.03814

Nagel, D., Weber, A., Lickert, B., & Stock, G. (2019). Dynamical coring of Markov state models. *J. Chem. Phys.*, *150*(9), 094111. https://doi.org/10.1063/1.5081767

Nagel, D., Weber, A., & Stock, G. (2020). MSMPathfinder: Identification of Pathways in Markov State Models. *J. Chem. Theory Comput.*, *16*(12), 7874–7882. https://doi.org/10.1021/acs.jctc.0c00774

Prinz, J.-H., Wu, H., Sarich, M., Keller, B., Senne, M., Held, M., Chodera, J. D., Schütte, C., & Noé, F. (2011). Markov models of molecular kinetics: Generation and validation. *J. Chem. Phys.*, *134*(17), 174105. https://doi.org/10.1063/1.3565032

Scherer, M. K., Trendelkamp-Schroer, B., Paul, F., Pérez-Hernández, G., Hoffmann, M., Plattner, N., Wehmeyer, C., Prinz, J.-H., & Noé, F. (2015). PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *J. Chem. Theory Comput.*, *11*, 5525–5542. https://doi.org/10.1021/acs.jctc.5b00743

The Pallets Projects. (2022). *Click: Python composable command line interface toolkit* (Version 8.1.*). https://palletsprojects.com/p/click

Wang, W., Cao, S., Zhu, L., & Huang, X. (2018). Constructing Markov state models to elucidate the functional conformational changes of complex biomolecules. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, *8*(1), e1343. https://doi.org/10.1002/wcms.1343