The Journal of Open Source Software

wedap: A Python package for weighted ensemble data analysis and plotting

Darian T. Yang [1,2,3] **and Lillian T. Chong** [1]

1 Department of Chemistry, University of Pittsburgh, USA **2** Department of Structural Biology, University of Pittsburgh, USA **3** Molecular Biophysics and Structural Biology Graduate Program, University of Pittsburgh and Carnegie Mellon University, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Biomolecules such as proteins, lipids, and nucleic acids are highly mobile and can have multiple interacting partners. Molecular dynamics (MD) simulations are a useful tool for studying these dynamic interactions by treating atoms and bonds as balls and springs; the movement of which can then be propagated using Newton's laws of motion. Even with this approximation, MD simulations work well to accurately capture small timescale dynamics for a variety of biomolecular systems (Karplus & Petsko, 1990). Due to current computational limitations, routine MD simulations typically only capture up to the μs or low ms timescale; however, some of the most interesting biological phenomena can take seconds or hours to occur (Zwier & Chong, 2010). To reach these longer timescales, a number of enhanced sampling strategies have been developed, including the weighted ensemble (WE) strategy (Huber & Kim, 1996; Zuckerman & Chong, 2017).

Compared to a standard MD simulation where the output is a single trajectory, the output from WE consists of multiple short trajectories that have an associated weight, which changes when the ensemble of trajectories are resampled at every iteration of the algorithm. The increased complexity of WE output makes standard analysis and plotting tools difficult to use since the changing weights must be accounted for during analysis and visualization steps. wedap is a Python package for simplifying weighted ensemble data analysis and plotting through a user-friendly command line interface, graphical user interface, and Python API to serve multiple levels of end-users.

Statement of need

Currently, the most cited software package for running WE simulations is the Weighted Ensemble Simulation Toolkit with Parallelization and Analysis (WESTPA) (Russo et al., 2022; Zwier et al., 2015). The WESTPA community is large and has had continued success using WE and WESTPA with many previous and ongoing projects (Saglam & Chong, 2019; Sztain et al., 2021; Zuckerman & Chong, 2017); a more comprehensive and continuously updated list of which can be found at https://westpa.github.io/westpa/publications.html. WE simulation data from WESTPA contains an ensemble of many short trajectories that vary in number and weight between each iteration. To address the difficulty of analyzing WE data, WESTPA comes with tools for generating and plotting probability distributions, among other analysis tools. While the native WESTPA-based plotting tools are functional, they have limitations that spurred motivation for the genesis of wedap. The WESTPA plotting tools are currently only accessible through the command line, and require two separate programs to generate a final figure from the output WESTPA dataset. This workflow is relatively straightforward for simple plotting examples, but quickly becomes complicated for more intricate plots, such as when using data other than the main progress coordinate dataset chosen for the WESTPA run.

When generating probability distributions of auxiliary datasets that are a part of the WESTPA output file or for advanced formatting of the plot, a separate Python script must be included for each WESTPA-based command. This increased involvement in analysis can be especially difficult for new users of WESTPA to analyze their output datasets beyond simple distributions. For advanced users, the WESTPA-based commands and their respective input Python files are often controlled using command line scripting, potentially leading to many intermediate files as you check the progress of a WESTPA simulation for numerous auxiliary datasets. Although the WESTPA plotting programs are written in Python, they currently do not have an easily accessible application programming interface (API) to fully customize the plot objects created.

In order to address these limitations, we designed wedap to provide a single, streamlined program for generating probability distributions and plots from WESTPA output datasets without the need for additional Python files. wedap is available through a command line program, a graphical user interface (GUI) using Gooey (Kiehl, 2019) and wxPython (Dunn & Pasanen, 2020), or directly through the Python API by importing wedap. This multimodality is useful for addressing a wider range of end-user skill sets. For those new to programming, the GUI requires much less computational background and can be accessed from the command line by simply typing wedap without any other arguments, the command line program allows for streamlined and convenient analysis that accommodates a wide range of users, and for advanced users who prefer to have everything within Python, the Python API provides appropriate classes to generate both probability distribution data (`wedap.H5_Pdist`) and plot objects (`wedap.H5_Plot`), which can then be expanded on if needed. The wedap Python API is flexible, allowing for easy coupling to existing Python workflows and to other Python packages. For example, the API can be used to make multi-panel figures or for plotting multiple datasets onto a single axis, opening up the figure making process to the full functionality of Python. All plotting methods in wedap are implemented using Matplotlib (Hunter, 2007), and the plot objects or probability distribution arrays are easily accessible as attributes of the generated objects, allowing easy adoption with other plotting packages such as seaborn (Waskom, 2021). All analysis operations in wedap are carried out using NumPy (Harris et al., 2020) for optimal memory usage and performance. h5py (Collette, 2014) was used to access and process the storage-optimized HDF5 files that are output from WESTPA.

Various examples of both basic and advanced Python scripts using wedap are available on the documentation webpage and the github repository. Examples are provided for guidance on general usage, as well as on interfacing with other Python packages such as gif (Humber, 2020) for creating animated plots or scikit-learn (Pedregosa et al., 2011) for accessing WE data as input to machine learning workflows. While we first implemented wedap strictly for generating probability distributions and plots, many additional features have been added and ongoing development is expected to continue. Additional features include being able to plot traces of individual pathways, plotting and adding a separately generated dataset seamlessly into the WESTPA output file, plotting probability distributions without specific basis states, and helper functions to extract raw data arrays and weights from the output WESTPA data file for easy pipelining to other functions. Overall, wedap is designed to provide a more accessible WE data analysis and plotting tool to the WESTPA simulation community with modularity that allows for flexibility of use and easy feature development. We note that wedap is already being used by different researchers and groups in the WE community and we expect to continue maintenance and development of wedap in parallel with the WESTPA software package.

Acknowledgements

We would like to thank everyone in the Chong lab for helpful discussions, testing out wedap in the early stages, and providing helpful feedback and suggestions. Feedback from and discussions with early users Jeremy Leung, Riccardo Solazzo, Anthony Bogetti, Marion Silvestrini, Rhea Abraham, and Hannah Piston were especially helpful. We would also like to thank the WESTPA developer community for helpful comments and suggestions. D.T.Y. was supported by a National Institutes of Health (NIH) Pittsburgh Aids Research Training (PART) program training grant T32AI065380 and a University of Pittsburgh Andrew Mellon Fellowship.

References

Collette, A. (2014). The h5py package is a pythonic interface to the HDF5 binary data format. In *GitHub repository*. GitHub. https://github.com/h5py/h5py

Dunn, R., & Pasanen, H. (2020). Gif: The matplotlib animation extension. In *GitHub repository*. GitHub. https://github.com/maxhumber/gif

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

Huber, G. A., & Kim, S. (1996). Weighted-ensemble Brownian dynamics simulations for protein association reactions. *Biophysical Journal*, *70*(1), 97–110. https://doi.org/10.1016/S0006-3495(96)79552-8

Humber, M. (2020). Gif: The matplotlib animation extension. In *GitHub repository*. GitHub. https://github.com/maxhumber/gif

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

Karplus, M., & Petsko, G. A. (1990). Molecular dynamics simulations in biology. *Nature*, *347*(6294), 631–639. https://doi.org/10.1038/347631a0

Kiehl, C. (2019). Gooey:turn (almost) any python 3 console program into a GUI application. In *GitHub repository*. GitHub. https://github.com/chriskiehl/Gooey

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Russo, J. D., Zhang, S., Leung, J. M. G., Bogetti, A. T., Thompson, J. P., DeGrave, A. J., Torrillo, P. A., Pratt, A. J., Wong, K. F., Xia, J., Copperman, J., Adelman, J. L., Zwier, M. C., LeBard, D. N., Zuckerman, D. M., & Chong, L. T. (2022). WESTPA 2.0: High-Performance Upgrades for Weighted Ensemble Simulations and Analysis of Longer-Timescale Applications. *Journal of Chemical Theory and Computation*, acs.jctc.1c01154. https://doi.org/10.1021/ACS.JCTC.1C01154

Saglam, A. S., & Chong, L. T. (2019). Protein-protein binding pathways and calculations of rate constants using fully-continuous, explicit-solvent simulations. *Chemical Science*, *10*(8), 2360–2372. https://doi.org/10.1039/C8SC04811H

Sztain, T., Ahn, S. H., Bogetti, A. T., Casalino, L., Goldsmith, J. A., Seitz, E., McCool, R. S., Kearns, F. L., Acosta-Reyes, F., Maji, S., Mashayekhi, G., McCammon, J. A., Ourmazd, A., Frank, J., McLellan, J. S., Chong, L. T., & Amaro, R. E. (2021). A glycan gate controls opening of the SARS-CoV-2 spike protein. *Nature Chemistry 2021 13:10*, *13*(10), 963–968. https://doi.org/10.1038/s41557-021-00758-3

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Zuckerman, D. M., & Chong, L. T. (2017). Weighted Ensemble Simulation: Review of Methodology, Applications, and Software. *Annual Review of Biophysics*, *46*(1), 43–57. https://doi.org/10.1146/annurev-biophys-070816-033834

Zwier, M. C., Adelman, J. L., Kaus, J. W., Pratt, A. J., Wong, K. F., Rego, N. B., Suárez, E., Lettieri, S., Wang, D. W., Grabe, M., Zuckerman, D. M., & Chong, L. T. (2015). WESTPA: An interoperable, highly scalable software package for weighted ensemble simulation and analysis. *Journal of Chemical Theory and Computation*, *11*(2), 800–809. https://doi.org/10.1021/ct5010615

Zwier, M. C., & Chong, L. T. (2010). Reaching biological timescales with all-atom molecular dynamics simulations. *Current Opinion in Pharmacology*, *10*(6), 745–752. https://doi.org/10.1016/j.coph.2010.09.008