Additional authors are inserted after DLD in alphabetical order.

Use footnote 6 and higher for affiliation.

alchemlyb: the simple alchemistry library

Zhiyi Wu[1*], **David L. Dotson**[2,3*], **David Mobley**[4], **Michael R. Shirts**[5], **and Oliver Beckstein**[2¶]

1 Exscientia, Oxford, UK **2** Arizona State University, Tempe, Arizona, USA **3** Datryllic LLC, Phoenix, Arizona, USA (present affiliation) **4** University of California Irvine, Irvine, California, USA **5** University of Colorado Boulder, Boulder, Colorado, USA ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

alchemlyb is an open-source Python software package for the analysis of alchemical free energy calculations, an important method in computational chemistry and biology, most notably in the field of drug discovery. Its functionality contains individual composable building blocks for all aspects of a full typical free energy analysis workflow, starting with the extraction of raw data from the output of diverse molecular simulation packages, moving on to data preprocessing tasks such as decorrelation of time series, using various estimators to derive free energy estimates from simulation samples, and finally providing quality analysis tools for data convergence checking and visualization. *alchemlyb* also contains high-level end-to-end workflows that combine multiple building blocks into a user-friendly analysis pipeline from the initial data input stage to the final results. This workflow functionality enhances accessibility by enabling researchers from diverse scientific backgrounds, and not solely computational chemistry specialists, to use *alchemlyb* effectively.

Statement of need

In the pharmaceutical sector, computational chemistry techniques are integral for evaluating potential drug compounds based on their protein binding affinity (Deng & Roux, 2009). Notably, absolute binding free energy calculations between proteins and ligands or relative binding affinity of ligands to the same protein are routinely employed for this purpose (Merz et al., 2010). The resultant estimates of these free energies are essential for understanding binding affinity throughout various stages of drug discovery, such as hit identification and lead optimization (Merz et al., 2010). Other free energies extracted from simulations are useful in solution thermodynamics, chemical engineering, environmental science, and material science.

Molecular simulation packages such as GROMACS (Abraham et al., 2015), Amber (Case et al., 2005), NAMD (Phillips et al., 2020), and GOMC (Nejahi et al., 2021) are used to run free energy simulations and many of these packages also contain tools for the subsequent processing of simulation data into free energies. However, there are no standard output formats and analysis tools implement different algorithms for the different stages of the free energy data processing pipeline. Therefore, it is very difficult to analyze data from different MD packages in a consistent manner. Furthermore, the native analysis tools do not always implement current best practices (Klimovich et al., 2015; Mey et al., 2020) or are out of date Overall, the coupling between data generation and analysis in most MD packages hinders seamless collaboration and comparison of results across different implementations of data generation for free energy calculations.

alchemlyb addresses this problem by focusing only on the data analysis portion of this process with the goal to provide a unified interface for working with free energy data generated from different MD packages. In an initial step data are read from the native MD package file formats and then organized into a common standard data structure, organized as a *pandas* `DataFrame` (McKinney, 2010). Functions are provided for pre-processing data by subsampling or decorrelation. Statistical mechanical estimators are available to extract free energies and thermodynamic expectations as well associated metrics of quality; these estimators are implemented as classes with the same API as estimators in scikit-learn (Buitinck et al., 2013; Pedregosa et al., 2011). *alchemlyb* implements modular building blocks to simplify the process of extracting crucial thermodynamic insights from molecular simulations in a uniform manner.

alchemlyb succeeds the widely-used but now deprecated `alchemical-analysis.py` tool (Klimovich et al., 2015), which combined pre-processing, free energy estimation, and plotting in a single script. `alchemical-analysis.py` was not thoroughly tested and hard to integrate into modern workflows due to its monolithic design, and only supported outdated Python 2. *alchemlyb* improves over its predecessor with a modular, function based design and thorough testing of all components using continuous integration. Thus, *alchemlyb* is a library that enables users to easily use well-tested building blocks within their own tools while additionally providing examples of complete end-to-end workflows. This innovation enables consistent processing of free energy data from diverse MD packages, facilitating streamlined comparison and combination of results.

Notably, *alchemlyb*'s robust and user-friendly nature has led to its integration into other automated workflow libraries such as BioSimSpace (Hedges et al., 2019) or MDPOW (Fan et al., 2020), demonstrating its accessibility and usability within broader scientific workflows and reinforcing its position as a versatile tool in the field of computational chemistry.

Implementation

Free energy differences are fundamental to understand many different processes at the molecular scale, ranging from the binding of drug molecules to their receptor proteins or nucleic acids through the partitioning of molecules into different solvents or phases to the stability of crystals and biomolecules. The calculation of such transfer free energies involves constructing two end states where a target molecule interacts with different environments. For example, in a solvation free energy calculation, at one state (the coupled state) it interacts with a solvent (in the case of hydration free energies, water), and the other where the ligand has no intermolecular interactions (the decoupled state), mimicking the transfer of a ligand at infinite dilution in the solvent at one end of the process and then ligand in the gas phase at the other. The solvation free energy is then obtained by calculating the free energy difference between these two end states. To achieve this, it is crucial to ensure sufficient overlap in phase space between the coupled and decoupled states, a condition often challenging to achieve.

Stratified alchemical free energy calculations have emerged as a de-facto standard approach whereby non-physical intermediate states are introduced to bridge between the physical end states of the process (Mey et al., 2020). In such free energy calculations, overlapping states are created by the introduction of a parameter λ that continuously connects the functional form (the Hamiltonian of the system) of the two end-states, resulting in a series of intermediate states each with a different λ value between 0 and 1 and with the physically realizable end states at $\lambda = 0$ and $\lambda = 1$. In general, N alchemical parameters are used to describe the alchemical transformation with a parameter vector $\vec{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_N)$, so that $\vec{\lambda} = (0, 0, \dots, 0)$ indicates the initial and $\vec{\lambda} = (1, 1, \dots, 1)$ the final state. The intermediate states are non-physical but required for converging the calculations. At each $\vec{\lambda}$-value (or "window"), the system configurations are sampled in the relevant thermodynamic ensemble, typically using MD or Monte Carlo (MC) simulations, while generating and accumulating free energy data discussed below. Estimators are then applied to these data to compute free energy differences between states, including the difference between the final and initial state, thus yielding the desired free energy difference of the physical process of interest.

Core design principles

alchemlyb is a Python library that seeks to make doing alchemical free energy calculations easier and less error prone. It includes functionality for parsing data from file formats of widely used simulation packages, subsampling these data, and fitting these data with an estimator to obtain free energies. Functions are simple in usage and pure in scope, and can be chained together to build customized analyses of data while estimators are implemented as a classes that follow the tried-and-tested scikit-learn API. General and robust workflows following best practices are also provided, which can be used as reference implementations and examples.

First and foremost, scientific code must be correct and we try to ensure this requirement by following best software engineering practices during development, close to full test coverage of all code in the library (currently 99%), and providing citations to published papers for included algorithms. We use a curated, public data set (*alchemtest*) for automated testing; code in *alchemtest* is published under the open source BSD-3 clause license while all data are included under an open license such as CC0 (public domain) or CC-BY (attribution required).

The guiding design principles are summarized as:

1. Use functions when possible, classes only when necessary (or for estimators, see (2)).
2. For estimators, mimic the object-oriented scikit-learn API as much as possible.
3. Aim for a consistent interface throughout, e.g. all parsers take similar inputs and yield a common set of outputs, using the `pandas.DataFrame` as the underlying data structure.
4. Have *all* functionality tested.

alchemlyb is published under the open source BSD-3 clause license.

Library structure

alchemlyb offers specific parsers in `alchemlyb.parsing` to load raw free energy data from various molecular simulation packages (GROMACS (Abraham et al., 2015), Amber (Case et al., 2005), NAMD (Phillips et al., 2020), and GOMC (Nejahi et al., 2021)) and provides a general structure for implementing parsers for other packages that are not yet supported. The raw data are converted into a standard format as a `pandas.DataFrame` and converted from the energy of the software to units of kT where $k = 1.380649 \times 10^{-23}$ J K^{-1} is Boltzmann's constant and T is the temperature at which the simulation was performed. Metadata such as T and the energy unit are stored in DataFrame attributes and propagated through *alchemlyb*, which enables seamless unit conversion with functions in the `alchemlyb.postprocessing` module. Two types of free energy data are considered: Hamiltonian gradients (`dHdl`, $dH/d\lambda$) at all lambda states, suitable for thermodynamic integration (TI) estimators (Kirkwood, 1935), and reduced potential energy differences between lambda states (`u_nk`, u_{nk}), which are used for free energy perturbation (FEP) estimators (Zwanzig, 1954).

Both types of estimators assume uncorrelated samples in order to give unbiased estimates of the uncertainties, which requires subsampling of the raw data. The `alchemlyb.preprocessing.subsampling` module provides tools for data subsampling based on autocorrelation times (J. D. Chodera et al., 2007; John D. Chodera, 2016) as well as simple slicing of the `dHdl` and `u_nk` DataFrames.

The two major classes of commonly used estimators are implemented in `alchemlyb.estimators`. Unlike other components of *alchemlyb* that are implemented as pure functions, estimators are implemented as classes and follow the well-known scikit-learn API (Buitinck et al., 2013) where instantiation sets the parameters (e.g., `estimator = MBAR(maximum_iterations=10000)`) and calling of the `fit()` method (e.g., `estimator.fit(u_nk)`) applies the estimator to the data and populates output attributes of the class; these results attributes are customarily indicated with a trailing underscore (e.g., `estimator.delta_f_` for the matrix of free energy differences between all states). In *alchemlyb*, TI (Paliwal & Shirts, 2011) and TI with Gaussian quadrature (Gusev et al., 2023) estimators are implemented in the TI category of estimators (module alchemlyb.estimators.TI). FEP category estimators (module alchemlyb.estimators.FEP) include Bennett Acceptance Ratio (BAR) (Bennett, 1976) and Multistate BAR (MBAR) (Shirts & Chodera, 2008), which are implemented in the *pymbar* package (Shirts & Chodera, 2008) and called from *alchemlyb*.

To evaluate the accuracy of the free energy estimate, *alchemlyb* offers a range of assessment tools. The error of the TI method is correlated with the average curvature (Pham & Shirts, 2011), while the error of FEP estimators depends on the overlap in sampled energy distributions (Pohorille et al., 2010). *alchemlyb* creates visualizations the smoothness of the integrand for TI estimators and the overlap matrix for FEP estimators, which can be qualitatively and quantitatively analyzed to determine the degree of overlap between simulated alchemical states, and suggest whether additional simulations should be run. For statistical validity, the accumulated samples should be collected from equilibrated simulations and *alchemlyb* contains tools for assessing (alchemlyb.convergence) and plotting (alchemlyb.visualisation) the convergence of the free energy estimate as a function of simulation time (Yang et al., 2004) and means to compute the "fractional equilibration time" (Fan et al., 2020) to detect potentially un-equilibrated data.

alchemlyb offers all these tools as a library for users to customize each stage of the analysis (Figure 1).

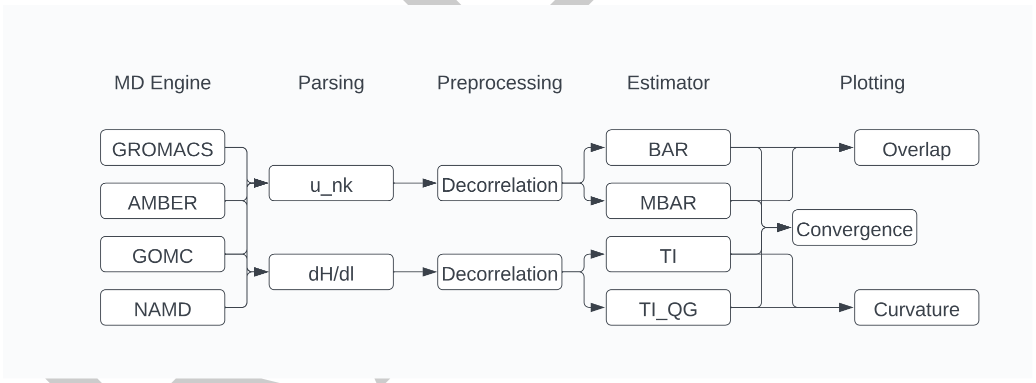

Figure 1: The building blocks of *alchemlyb*. Raw data from simulation packages are parsed into common data structures depending on the free energy quantities, pre-processed, and processed with a free energy estimator. The resulting free energy differences are analyzed for convergence and plotted for quality assessment.

Workflows

The building blocks are sufficient to compute free energies from alchemical free energy simulations and assess their reliability. *alchemlyb* also provides a structure to combined the building blocks into full end-to-end workflows (module alchemlyb.workflows). As an example, the ABFE workflow for absolute binding free energy estimation reads in the raw input data and performs decorrelation, estimation, and quality plotting of the estimate. It can directly estimate quantities such as solvation free energies and makes it easy to calculate more complex quantities such as absolute binding free energies (as the difference between the solvation free energy of the ligand in water and the solvation free energy of the ligand in the protein's binding pocket).

Acknowledgements

add relevant funding information

Some work on alchemlyb was supported by grants from the National Institutes of Health (Award No R01GM118772 to O.B.) and the National Science Foundation (award ACI-1443054

If listed as AUTHOR remove from ACKNOWLEDGEMENTS

to O.B.). The sponsors were not involved in any aspects of the research or the writing of the manuscript.

Ian Kenney, Shuai Liu, Travis Jensen, Bryce Allen, Dominik Wille, Victoria Lim, Hyungro Lee, Mohammad S. Barhaghi, Alexander Schlaich, Jérôme Hénin, Thomas T. Joseph, Irfan Alibay, Pascal Merz, Domenico Marson, and Haoxi Li contributed code to *alchemlyb*.

new authors: add your initials after DD under "contributed new features"

Author contributions

D.D., M.R.S., D.M., and O.B. designed the project. Z.W., D.D., contributed to new features. Z.W., D.D., O.B. maintained the code base. Z.W., D.D., M.R.S, O.B. wrote the manuscript.

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. https://doi.org/10.1016/j.softx.2015.06.001

Bennett, C. H. (1976). Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, *22*(2), 245–268.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Case, D. A., Cheatham, T. E., 3rd, Darden, T., Gohlke, H., Luo, R., Merz, K. M., Jr, Onufriev, A., Simmerling, C., Wang, B., & Woods, R. J. (2005). The Amber biomolecular simulation programs. *J Comput Chem*, *26*(16), 1668–1688. https://doi.org/10.1002/jcc.20290

Chodera, John D. (2016). A simple method for automated equilibration detection in molecular simulations. *Journal of Chemical Theory and Computation*, *12*(4), 1799–1805. https://doi.org/10.1021/acs.jctc.5b00784

Chodera, J. D., Swope, W. C., Pitera, J. W., Seok, C., & Dill, K. A. (2007). Use of the weighted histogram analysis method for the analysis of simulated and parallel tempering simulations. *J Chem Theory Comput*, *3*(1), 26–41. https://doi.org/10.1021/ct0502864

Deng, Y., & Roux, B. (2009). Computations of standard binding free energies with molecular dynamics simulations. *J Phys Chem B*, *113*(8), 2234–2246. https://doi.org/10.1021/jp807701h

Fan, S., Iorga, B. I., & Beckstein, O. (2020). Prediction of octanol-water partition coefficients for the SAMPL6-$\log P$ molecules using molecular dynamics simulations with OPLS-AA, AMBER and CHARMM force fields. *Journal of Computer-Aided Molecular Design*, *34*, 543–560. https://doi.org/10.1007/s10822-019-00267-z

Gusev, F., Gutkin, E., Kurnikova, M. G., & Isayev, O. (2023). Active learning guided drug design lead optimization based on relative binding free energy modeling. *Journal of Chemical Information and Modeling*, *63*(2), 583–594. https://doi.org/10.1021/acs.jcim.2c01052

Hedges, L. O., Mey, A. S. J. S., Laughton, C. A., Gervasio, F. L., Mulholland, A. J., Woods, C. J., & Michel, J. (2019). BioSimSpace: An interoperable Python framework for biomolecular simulation. *Journal of Open Source Software*, *4*(43), 1831. https://doi.org/10.21105/joss.01831

Kirkwood, J. G. (1935). Statistical mechanics of fluid mixtures. *The Journal of Chemical Physics*, *3*(5), 300–313.

Klimovich, P. V., Shirts, M. R., & Mobley, D. L. (2015). Guidelines for the analysis of free energy calculations. *J Comput Aided Mol Des*, *29*(5), 397–411. https://doi.org/10.1007/s10822-015-9840-9

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Merz, K. M., Jr, Ringe, D., & Reynolds, C. H. (2010). *Drug design: Structure-and ligand-based approaches*. Cambridge University Press.

Mey, A. S. J. S., Allen, B., Macdonald, H. E. B., Chodera, J. D., Kuhn, M., Michel, J., Mobley, D. L., Naden, L. N., Prasad, S., Rizzi, A., Scheen, J., Shirts, M. R., Tresadern, G., & Xu, H. (2020). Best practices for alchemical free energy calculations. *Living Journal of Computational Molecular Science*, *2*(1), 18378. https://doi.org/10.33011/livecoms.2.1.18378

Nejahi, Y., Barhaghi, M. S., Schwing, G., Schwiebert, L., & Potoff, J. (2021). Update 2.70 to "GOMC: GPU optimized Monte Carlo for the simulation of phase equilibria and physical properties of complex fluids." *SoftwareX*, *13*, 100627. https://doi.org/10.1016/j.softx.2020.100627

Paliwal, H., & Shirts, M. R. (2011). A benchmark test set for alchemical free energy transformations and its use to quantify error in common free energy methods. *J Chem Theory Comput*, *7*(12), 4115–4134. https://doi.org/10.1021/ct2003995

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.

Pham, T. T., & Shirts, M. R. (2011). Identifying low variance pathways for free energy calculations of molecular transformations in solution phase. *J Chem Phys*, *135*(3), 034114. https://doi.org/10.1063/1.3607597

Phillips, J. C., Hardy, D. J., Maia, J. D. C., Stone, J. E., Ribeiro, J. V., Bernardi, R. C., & al., et. (2020). Scalable molecular dynamics on CPU and GPU architectures with NAMD. *The Journal of Chemical Physics*, *153*(4), 044130. https://doi.org/10.1063/5.0014475

Pohorille, A., Jarzynski, C., & Chipot, C. (2010). Good practices in free-energy calculations. *J Phys Chem B*, *114*(32), 10235–10253. https://doi.org/10.1021/jp102971x

Shirts, M. R., & Chodera, J. D. (2008). Statistically optimal analysis of samples from multiple equilibrium states. *J Chem Phys*, *129*(12), 124105. https://doi.org/10.1063/1.2978177

Yang, W., Bitetti-Putzer, R., & Karplus, M. (2004). Free energy simulations: Use of reverse cumulative averaging to determine the equilibrated region and the time required for convergence. *J Chem Phys*, *120*(6), 2618–2628. https://doi.org/10.1063/1.1638996

Zwanzig, R. W. (1954). High-temperature equation of state by a perturbation method. I. Nonpolar gases. *The Journal of Chemical Physics*, *22*(8), 1420–1426.