A Suite of Tutorials for the BioSimSpace Framework for Interoperable Biomolecular Simulation [Article v1.0]

Lester O. Hedges[1,2*], Sofia Bariami[3†], Matthew Burman[2], Finlay Clark[3], Benjamin P. Cossins[4], Adele Hardie[3], Anna M. Herz[3], Dominykas Lukauskis[5], Antonia S.J.S. Mey[3], Julien Michel[2,3*], Jenke Scheen[3‡], Miroslav Suruzhon[4], Christopher J. Woods[1], Zhiyi Wu[4]

[1]Advanced Computing Research Centre, University of Bristol, UK; [2]OpenBioSim Community Interest Company, UK; [3]EaStCHEM School of Chemistry, University of Edinburgh, UK; [4]Exscientia Plc., Oxford, UK; [5]Department of Chemistry and Institute of Structural and Molecular Biology, University College London, UK

This LiveCoMS document is maintained online on GitHub at https://github.com/OpenBioSim/biosimspace_tutorials; to provide feedback, suggestions, or help improve it, please visit the GitHub repository and participate via the issue tracker.

This version dated December 14, 2023

Abstract This tutorial serves as a getting-started guide for BioSimSpace (BSS), an interoperable molecular simulation framework, that allows simulations with different sets of molecular dynamics software packages. This tutorial will cover four main use cases for BioSimSpace. The introductory tutorial introduces the basic structure of BioSimSpace, how to use the API to access functionality, and how to write code for setting up and running standard molecular dynamics simulations. Three advanced use cases of BSS are then provided, describing how to set up and run a funnel metadynamics simulation, steered molecular dynamics, and relative or absolute alchemical binding free energy calculations.

***For correspondence:**
julien.michel@ed.ac.uk (JM); lester@openbiosim.org (LH)
Present address: †Cresset Litlington SG8 0SS, Cambridgeshire, UK; ‡Open Molecular Science Foundation, 200 B Street, Suite F, Davis CA 95616, USA

1 Introduction

BioSimSpace (BSS) [1] (http://openbiosim.biosimspace.org) is a software that allows the setup, simulation, and analysis of different types of molecular dynamics (MD) simulation methodologies with different software packages. The software provides a common API in Python for the following MD software packages: AMBER [2], GROMACS [3], NAMD [4], OpenMM [5] and has built-in analysis of simulation through MDAnalysis [6], MDTraj [7], alchemlyb [8]. BSS was originally developed by the academic consortium CCPBioSim (https://www.ccpbiosim.ac.uk) and is currently maintained by the OpenBioSim Community Interest Company (https://www.openbiosim.org).

2 Prerequisites and Scope

Received: 20 July 2023
Accepted: 13 December 2023

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

2.1 Background knowledge

To get the most out of this tutorial the reader should have at least Intermediate proficiency in Python and be familiar with Jupyter notebooks. For the introductory set of tutorials a background in biomolecular dynamics simulations is expected [9], and if you are unfamiliar with this topic the best practices guide on foundations in biomolecular simulations is a good starting point [10]. Tutorial 2 on funnel metadynamics expects some background understanding of funnel metadynamics; a good starting point for studying this topic can be found in ref [11]. Tutorial 3, expects knowledge on steered MD and collective variables (CVs). If you are unfamiliar with these topics Israelewitz et al. provides a good introduction to steered MD [12], while a chapter in *Biomolecular Simulations* by Bussi and Tribello gives a practical overview on how CVs are used by PLUMED[13]. Tutorial 4 covers how to use BSS to run alchemical free energy calculations. The best-practice guide by Mey et al. for alchemical free energy calculations provides a broad introduction and overview of this method [14].

2.2 Software/system requirements and installation

Unless otherwise mentioned the tutorials have been tested with the following software packages.

- BioSimSpace release (2023.3.0 or more recent) https://biosimspace.openbiosim.org
- Gromacs 2023.1 https://www.gromacs.org/
- AmberTools 23 https://ambermd.org/AmberTools.php
- PLUMED 2.9.0 (required for Tutorials 2 and 3) https://www.plumed.org/
- cinnabar 0.3.0 (required for Tutorial 4) https://github.com/OpenFreeEnergy/Cinnabar

In addition users that have a Amber22 software licence may wish to use the MD engine pmemd as a replacement for the slower sander MD engine available with AmberTools.

- Amber 22 (Used in Tutorials 3 and 4) https://ambermd.org/AmberMD.php

We recommend following the installation instructions available at https://github.com/OpenBioSim/biosimspace_tutorialsto install all the required dependencies. OpenBioSim also provides access to a complete BSS environment with all required dependencies to run the tutorial suite via a free Jupyter Hub server hosted at https://www.openbiosim.org/demos. Login requires a valid GitHub user account.

3 Tutorial suite

The tutorial suite consists of several Jupyter notebooks and can be found here: https://github.com/OpenBioSim/biosimspace_tutorials. The individual tutorials available in the suite are summarized below, and the reader is encouraged to consult the notebooks for full details.

3.1 Tutorial 1: Introduction to BioSimSpace

This tutorial consists of five separate notebooks.

The first notebook 01-introduction describes what functionality is currently available in BioSimSpace; what are the key concepts behind BioSimSpace, such as the use of a Sire system object to describe a generic molecular system; how interoperability between packages is achieved through a library of file converters; what steps are taken to preserve the topology of a molecular system after it has been passed to various third-party tools.

The second notebook 02-molecularsetup provides an example of how to use BioSimSpace to set up a molecular system ready for simulation. Starting from a molecular topology in the form of a Protein Data Bank format file, the notebook teaches how to parameterize molecules using different molecular force fields, then solvate them using various water models before exporting the solvated topologies in a chosen file format.

The third notebook 03-molecular-dynamics describes how to use BioSimSpace to configure and run some basic molecular dynamics simulations. This notebook introduces the concept of BioSimSpace.Protocol to codify a shareable, re-usable and extensible simulation protocol. The concept of BioSimSpace.Process is then introduced to provide functionality for configuring and running processes with several common molecular dynamics engines. The process is then illustrated with code that implements interactive molecular dynamics simulations in the notebook. Finally, BioSimSpace.Trajectory is introduced as a wrapper for the python software MDTraj and MDAnalysis to facilitate trajectory analyzes.

The fourth notebook 04-writing-nodes introduces the concept of nodes as interoperable workflow components. Nodes are robust and portable Python scripts that typically do a small, well-defined piece of work. All inputs and outputs from the node are validated and the node is written in such a way that it is independent of the underlying software packages, i.e. the same script can work with a range of different packages. In addition, nodes are aware of the environment in which they are run, so can be used interactively, from the command-line, or within a workflow engine.

The fifth notebook 05-running-nodes describes how BioSimSpace nodes can be exported as regular python

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

scripts and executed in a variety of environments. This allows users to prototype BioSimSpace nodes in an interactive Jupyter notebook, and then deploy the same script without having to insert additional code. BioSimSpace nodes can be currently called from the command line and can also be imported within a BioSimSpace script, enabling the construction of complex workflows by reusing libraries of nodes. Nodes can also generate common workflow language wrappers, allowing BSS nodes to be plugged into any workflow engine that supports this standard.

3.2 Tutorial 2: funnel metadynamics

3.2.1 Introduction

Funnel metadynamics (*fun-metaD*) is a molecular dynamics-based method that calculates the absolute binding free energy (ABFE) between a small organic ligand and a protein. It uses the enhanced sampling method metadynamics. To best separate the bound and unbound phases, as well as increase the rate of convergence of the binding free energy, the exploration of the ligand in 3D space is limited by funnel-shaped restraints.

Readers that lack familiarity with metadynamics may wish to study first a simple BioSimSpace metadynamics tutorials for a alanine dipeptide molecule in the gas phase.

fun-metaD was originally implemented by Limogelli et al. [11]. This tutorial is based on the implementation described by Rhys et al [15], and Saleh et al [16]. The main difference between the original and later implementations is the functional form of the funnel restraints: the original *fun-metaD* relied on a cone and a cylinder joined to make a funnel using a step function, while the new implementation uses a single sigmoid function. The Limogelli implementation also requires the protein to be realigned with a reference structure to keep the funnel strictly in place over the binding site, which negatively affects performance. The implementation described here allows the funnel to move with the protein.

A challenge with using *fun-metaD* for ABFE calculations is that it can be difficult to select optimised funnel parameters, and tedious to write multiple input files for PLUMED. To facilitate use of *fun-metaD* by non-experts BioSimSpace implements automated setup algorithms for selecting funnel parameters, and producing input files.

By the end of this tutorial, you should understand:

- The theory that underpins *fun-metaD* calculations.
- How to set up *fun-metaD* simulations and how to visualise the funnel restraints.
- How to analyse the results of a *fun-metaD* simulation.

3.2.2 Theory

Metadynamics is an enhanced sampling method that biases a simulation along a chosen set of reaction coordinates, or as MetaD practitioners call them, collective variables (CVs). This bias is deposited at defined time intervals and takes the shape of a Gaussian potential. Investigation of drug binding should involve at least one CV, the distance from the drug molecule to the protein, where the distance between them can be biased, causing the drug to unbind. However, that single distance is degenerate, meaning many different configurations of the drug in 3D space will not be described by that single distance. It also involves the exploration of a very large volume, hindering convergence.

Fun-metaD gets around both of these problems, restricting the exploration by using funnel-shaped restraints. The restraints are defined with a 2D coordinates system based on the two CVs - 'projection' and 'extent'. The first CV measures the distance along the axis-vector defined by the coordinates of the points P_0 and P_X in the laboratory frame. The second CV measures the distance along a second axis-vector orthogonal to the first axis-vector. See Figure 1.

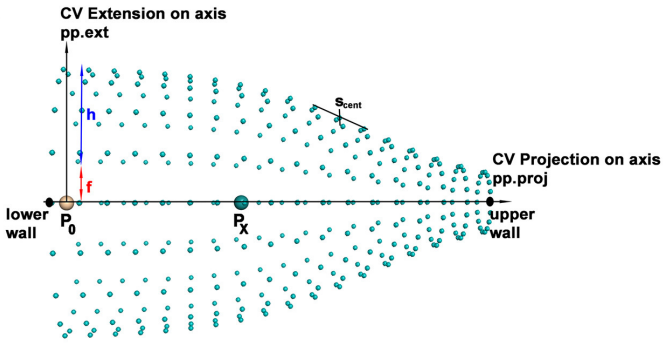

Figure 1. Visualisation of the funnel restraints defined using the CV projection axis and the CV extent axis.

The funnel parameters h and f controls the maximal and minimal width of the funnel along the extent axis respectively. The smoothness of the transition between maximal and minimal widths along the projection axis is controlled by the parameters b and x that together control the location and steepness of the inflection point located at S_{cent} in fig 1. All together these parameters define the radius S of the funnel at a given distance i along the projection axis.

$$S(i) = h\left(\frac{1}{1 + b^{b(i-x)}}\right) + f, \tag{1}$$

Clearly, there is still some degeneracy in the CVs - the plane perpendicular to the projection axis is a one-dimensional representation of a two-dimensional space. However, this is a good compromise between having sufficient accuracy for describing the binding of a ligand and the

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

tolerable simulation slowdown of using only two CVs.

In order to have a good separation between the bound and unbound phases for the ABFE estimations, the funnel needs to point *out*, with the narrow end in the solvent, and away from any protein residues. The BSS automated funnel assignment code does this generally well. Typically the vectors picked for defining the P_0 and P_1 points offer an unobstructed exit path for the ligand. It is nonetheless still a good idea to visually check the proposed funnel, especially for new protein systems. This can be done through use of the funnel visualisation functionality within BSS (see below).

The size of the funnel radius has to be determined on a case-by-case basis. The metadynamics code tracks the center of mass the ligand, therefore ligand atoms are able to sample positions outside the visualised funnel, provided the center of mass remains inside the funnel. the volume that the small molecule will explore is much larger than implied by the visualization of the funnel. There is usually only one binding site and the funnel should enclose only it, excluding other protein features, by setting a small value of *h*. This helps accelerate convergence by preventing the ligand from exploring irrelevant regions in the free energy surface (FES).

3.2.3 Setting up funnel metadynamics simulations

The first tutorial notebook 01-bss-fun-metad-setup shows how to set up a BioSimSpace system, parameterizing the protein and the ligand, as well as defining the simulation box, adding water and ions. A funnel is then defined and visualized using NGLview. Finally, directories for the *funmetaD* simulation are setup and a short 10 ps simulation is run for illustrative purposes.

The notebook uses as example a fully solvated HSP90 protein-ligand system originally setup from PDBID:2WI2. Figure 2 depicts a rendered funnel overlayed on the HSP90 protein-ligand system in the Jupyter Notebook. The funnel vector is clearly pointing out into the solvent, with no protein residues blocking the way. The default radius is sufficient to encompass the binding site, excluding the rest of the protein.

To obtain converged free energies we recommend simulations of the order of 500 - 2000 ns sampling time. This is best done out of a notebook. For convenience we also provide with this tutorial as companion resource an LSF submission script that re-implements the notebook functionality in a set of BioSimSpace nodes.

3.2.4 Analysing funnel metadynamics simulations

The second notebook of this tutorial 02-bss-fun-metad-analysis describes how to analyze a funnel metadynamics simulation. The precision of the ABFE estimate derived from a funnel metadynamics run is linked to the convergence of

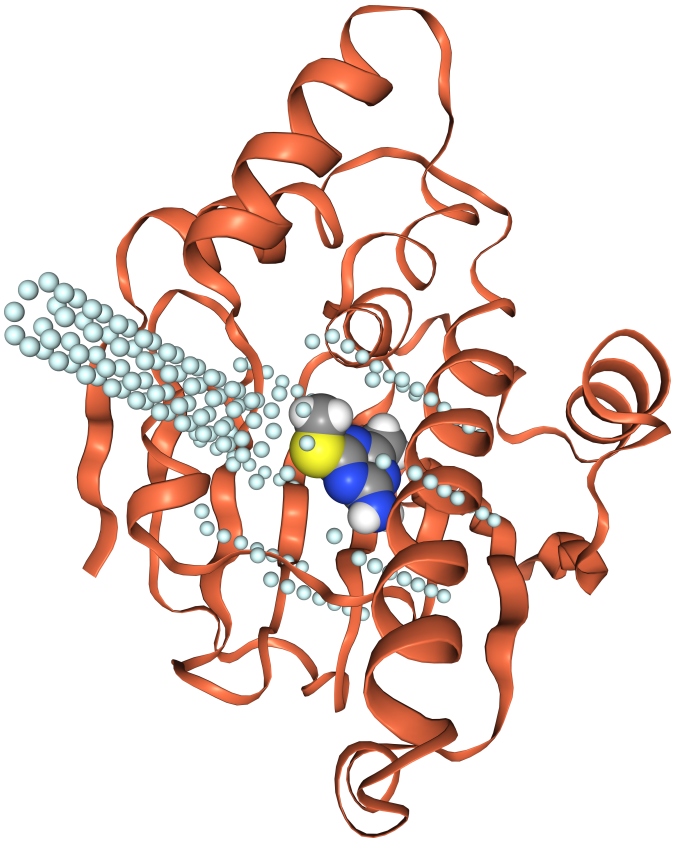

Figure 2. Visualisation of funnel restraints autogenerated by BioSimSpace for a HSP90 protein-ligand system.

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

the free energy profile along the collective variables that define the funnel. The notebook uses provided sample trajectories to show how the range of CV values sampled during a trajectory and two-dimensional free energy surfaces can be plotted with the help of BSS.Notebook. The notebook also shows how to compute funnel correction terms using BSS, and how to perform convergence analysis of the absolute binding free energies for different trajectories. Selected visualizations generated by the notebook are depicted in Figure 3.

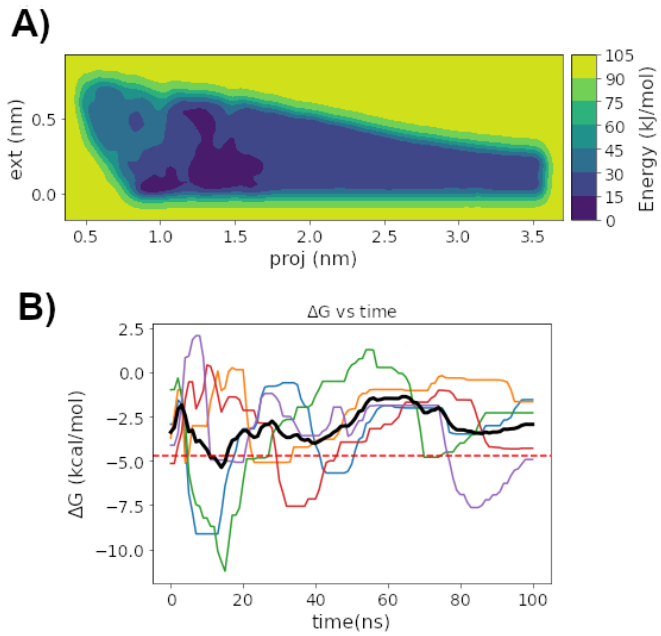

Figure 3. Selected analyses of funnel metadynamics simulations of the HSP90 protein-ligand system. A) Reconstructed two-dimensional free energy profile from a sample 100 ns trajectory. B) Convergence plots for five replicates of a 100-ns trajectory. The bold black line denotes the mean of the replicates, and the dashed red line the experimental estimate of the absolute binding free energy of the ligand.

3.3 Tutorial 3: Steered molecular dynamics

3.3.1 Introduction

Many relevant biological processes, such as transmembrane permeation or transitions between active and inactive protein conformations, occur on a timescale of microseconds to seconds [17–19]. However, even with GPU acceleration, the timescales accessible via MD simulations are only a few hundred ns/day [20]. One of the methods to get around this limitation is steered molecular dynamics (sMD). sMD involves applying a harmonic restraint to bias the system towards a conformation defined through one or more collective variables (CVs):

$$V(\vec{s}, t) = \frac{1}{2}\kappa(t)(\vec{s} - \vec{s}_0(t))^2, \quad (2)$$

where κ is the force constant, $\vec{s}_0$ is the expected CV value at a specific timestep, and $\vec{s}$ is the actual CV value at that timestep [12, 21].

This section of the tutorial summarizes how to use BioSimSpace to set up and run sMD simulations. BSS prepares input files for PLUMED, which is the software that works together with MD engines such as AMBER and GROMACS to add the restraint in eq 2.

We use protein tyrosine phosphatase 1B (PTP1B) as the system of choice for this tutorial. It is a negative regulator of insulin signalling [22], and is an attractive target for type II diabetes [23]. The function of PTP1B depends on the conformation of its WPD loop, which can be closed (active) or open (inactive) (Figure 4). The WPD loop of PTP1B opens and closes on a μs timescale [18], and therefore this transition is not observed on conventional computational timescales.

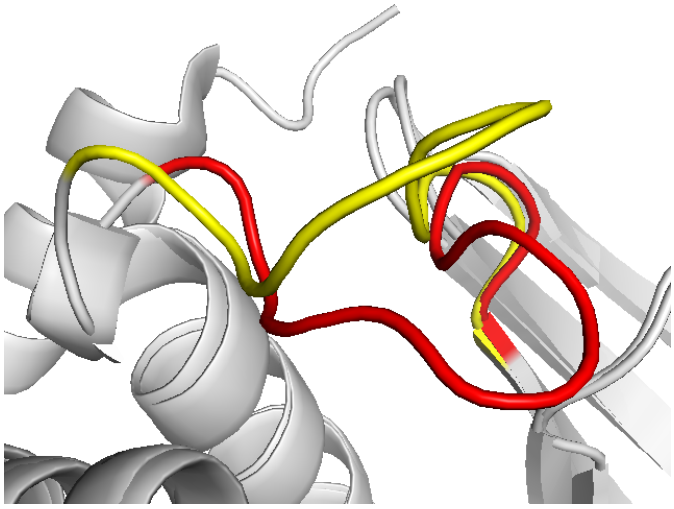

Figure 4. The WPD loop of PTP1B, in two conformations: open (yellow, PDB ID: 2HNP) and closed (red, PDB ID: 1SUG).

3.3.2 Running sMD using BioSimSpace

The first notebook of this tutorial 01-setup-sMD describes how to set up a steered MD simulation with BioSimSpace. The notebook illustrates the use of BSS.Metadynamics.CollectiveVariable.RMSD to define a collective variable that enforces a conformational change of the 'WPD' loop in the enzyme PTP1B. Next BSS.Protocol.Steering is used to specify a steering schedule alongside the RMSD CV. The notebook illustrates how input files for the *gmx*, *sander* or *pmemd* MD engines can be subsequently prepared. The notebook also shows how a more complex steering schedule that combines multiple CVs can be written.

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

For production simulations, we recommend long sMD simulations to minimize the strength of the bias that needs to be applied to enforce the desired conformational change by the end of the steered MD simulation. Optimizing the steered MD schedule parameters requires trial and error. For convenience, we provide simple Python scripts with a command line interface to execute steered MD runs and scan schedule parameters for two specified sMD protocols (a single CV and a multiple CV example). We also provide sample slurm and LSF submission scripts to deploy the BSS steered MD scripts in different HPC environments.

3.3.3 sMD trajectory analysis

The second notebook of this tutorial 02-trajectory-analysis describes how to analyze data generated by a steered MD run. As the sMD simulation is run, the CV values are saved to a **COLVAR** file. It can be plotted to assess whether the sMD simulation has been successful. An example is shown in Figure 5.

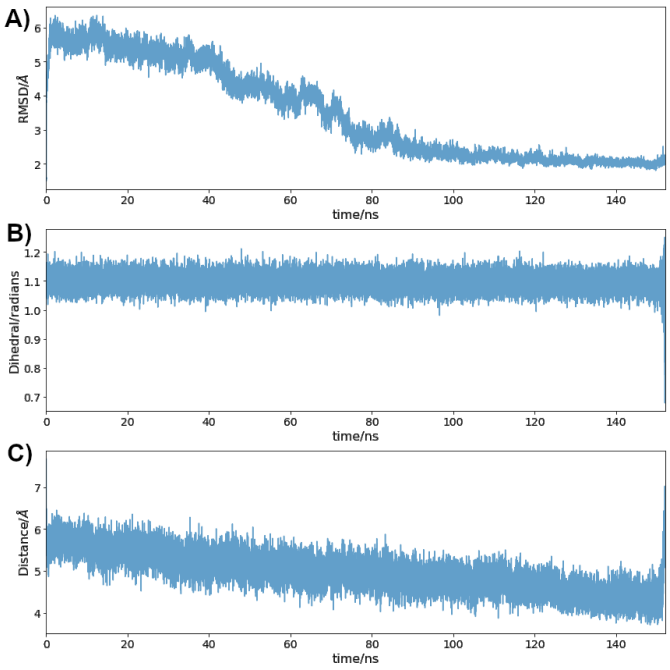

Figure 5. Evolution of Collective Variables throughout an sMD simulation. A) RMSD CV measuring distance to closed WPD loop conformation. B) Torsional angle CV measuring the Chi1 angle of residue Tyr 152. C) CV measuring the distance between C_γ atoms in residues Phe196 and Phe280. As the simulation progresses, the WPD loop RMSD is gradually lower (i.e. the loop is adopting a closed loop conformation).

The notebook also illustrates a "failed" steered MD trajectory, where the steering duration and force were insufficient to reach the target CV value.

3.3.4 Example application - combining steered MD with Markov State Modeling

While the information provided here focuses on running sMD simulations with BSS, there are multiple potential applications, such as studying membrane permeability [19] or ligand residence time[24]. Here we briefly highlight one application of sMD simulations enabled by BioSimSpace in the AMMo software project. AMMo ("Allostery in Markov Models") was developed to evaluate the allosteric effects of protein mutations or ligand binding by combining sMD with Markov State Models (MSMs). MSMs are used to give the probability of protein conformations and therefore can be used to model how a ligand affects the conformation ensemble of a target (e.g. whether the presence of a ligand decreases the active state probability and therefore is an allosteric inhibitor). There is a lot to consider when building MSMs, and the method is not covered in this tutorial. AMMo uses the Python library PyEMMA was to implement MSMs. Extensive examples and documentation for PyEMMA are available elsewhere [25]. The integration of sMD with MSM in this allosteric modulation prediction workflow is illustrated in Figure 6. The notebook 02-trajectory-analysis shows how a sMD trajectory can be sampled to extract a range of protein conformations suitable for inputs to an MSM workflow. Hardie et al. report a detailed study of allosteric modulators of PTP1B using this sMD/MSM methodology [26] and notebooks for the PTP1B case study are available on the GitHub page for AMMo.

3.4 Tutorial 4: Alchemical free energy calculations

3.4.1 Introduction

Computational chemists can support the study of structure-activity relationships in medicinal chemistry by making computer models that can predict the binding affinity of ligands to proteins. Alchemical free energy (AFE) methods are a popular class of methodologies to do so. Some introductory reading is recommended [14, 27–29].

This tutorial covers the basic principles of alchemical free energy calculations with BioSimSpace; how to setup, simulate, and analyze alchemical Relative Binding Free Energy (RBFE) calculations for congeneric series of protein-ligand complexes; how to set up and analyze alchemical ABFE calculations of a ligand bound to a protein. The notebooks prompt the readers to complete a series of exercises that typically involve completing cells to test their understanding of the material presented.

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

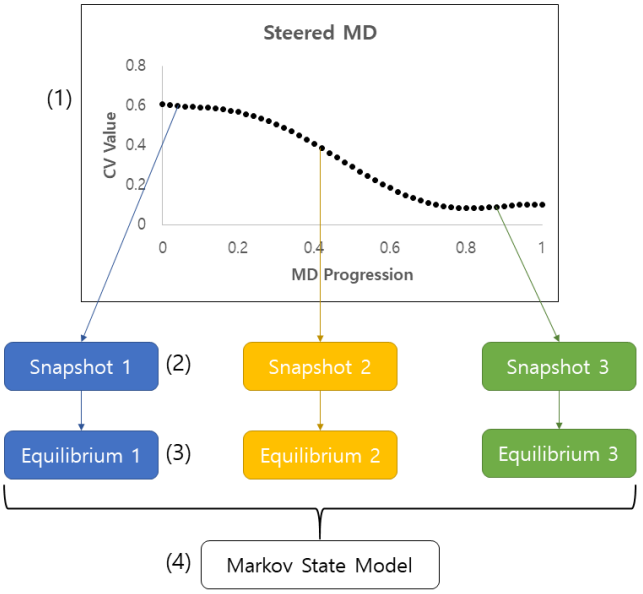

Figure 6. The steps used for enhanced sampling methods to gather data for statistical analysis of protein conformation ensemble. (1) Run steered MD along some collective variable (CV); (2) Extract snapshots that evenly sample available conformational space; (3) Run equilibrium MD simulations using extracted coordinates as seeds; (4) construct an MSM using trajectory data from step 3.

3.4.2 Setting up AFE calculations using BioSimSpace

The first notebook of this tutorial alchemical-introduction introduces the basic functionality available in BioSimSpace to implement AFE calculations. The notebook first introduces the functionality of the BSS.Align module. BSS.Align implements a variety of mapping algorithms based on maximum common substructure searches between a supplied pair of molecules to generate a 'merged' molecule used to describe an alchemical transformation. This is used to generate a merged molecule that describes the alchemical transformation of ethane into methanol. Next, the use of BSS.Protocol.FreeEnergy is illustrated using the relative hydration free energy of ethanol to methane as an example, using the AFE engines *SOMD* [30] or *mdrun*. The AFE simulations can in principle be executed directly from the notebook as a set of independent MD simulations run serially, but this is often too slow to be practical. The next section describes protocols to submit calculations in parallel on HPC resources. Finally, the notebook describes the setup of the relative binding free energy calculation of benzene to o-xylene bound to the protein T4 lyzozyme mutant.

The notebook then describes the use of BSS.FreeEnergy.Relative to process the completed AFE simulations for two legs of a thermodynamic cycle using MBAR or TI to obtain a free energy estimate. The resulting free energy changes are subtracted to yield a relative binding free energy for o-xylene to benzene. Finally, the plotting of overlap matrices is demonstrated to assess the reliability of a computed free energy change. The functionality covered by this notebook is illustrated in Figure 7.

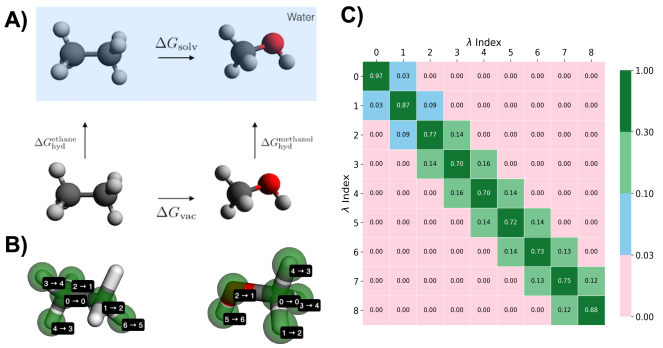

Figure 7. **A)** Thermodynamic cycle used to compute the relative hydration free energy of ethane to methanol. **B)** Visualisation of the atom mappings between ethane and methanol generated by *BSS.Align*. **C)** Visualisation of overlap matrix generated by *BSS.Notebook.plotOverlapMatrix* to help assess the reliability of a calculated free energy change.

3.4.3 RBFE calculation pipelines

The second notebook of this tutorial 01-setup-rbfe.ipynb shows how to design an RBFE campaign for a congeneric series of protein-ligand complexes. This is illustrated using a dataset of ligands for the protein TYK2, taken from the benchmark set of Wang et al. [31] First, a drop-down menu is presented to the user to enable the selection of a variety of configuration settings (such as the choice of forcefields to use for the ligands, the protein, or the FEP engine to select). In BSS 2023.3.0 the FEP engines *SOMD* and *mdrun* (from the GROMACS software suite) are supported. An experimental feature branch that implements support for RBFE calculations with the MD engine pmemd from the AMBER software suite is also available.

Next, BSS.Align.generateNetwork is used to interface with the LOMAP software [32] to propose a network of relative transformations that span the provided ligand dataset. The notebook illustrates interactive plotting of the proposed network and how to make manual adjustments such as adding or deleting edges or inserting new ligands into the network. Once the user is satisfied with the chosen FEP network, setup instructions are saved to disk.

Processing the entire TYK2 dataset involves setting up and running several hundred MD simulations of solvated ligands and protein-ligand complexes. This would be impractically slow if executed from a notebook on a single workstation. For convenience, we provide a sample slurm submission script that processes the setup, simulation and analysis of the entire network constructed by the RBFE setup notebook

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

on an HPC environment. This script may be adjusted for deployment on different slurm clusters or as reference for the implementation of the execution model on different schedulers. The functionality covered by this notebook is illustrated in Figure 8.

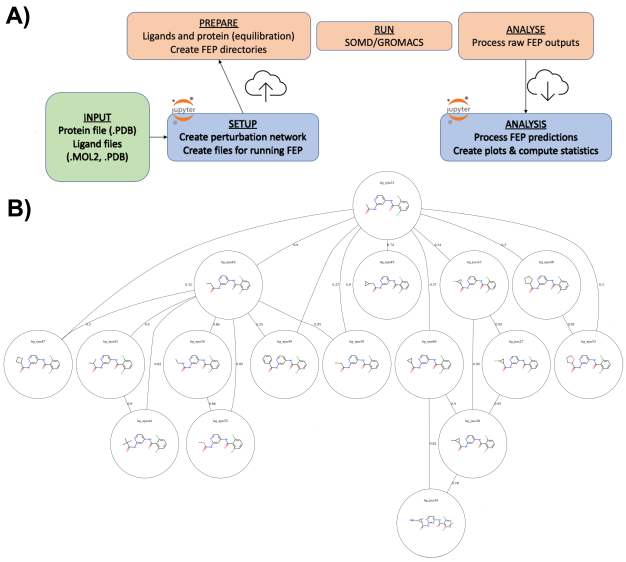

Figure 8. A) Schematic of the RBFE pipeline in this tutorial. Whereas blue boxes represent notebooks run on a local machine, orange boxes represent Python scripts run sequentially on a computing cluster. **B)** Perturbation network proposed by *BSS.Align.generateNetwork* for a dataset of TYK2 ligands.

The third notebook 02-analysis-rbfe provides a walkthrough of the analysis of the processed RBFE network. The RBFE network is first visualized using *NetworkX*. Next, mean relative binding free energies are evaluated for each edge by averaging the results from all replicates available for each edge. The standard error of the mean is used as an estimate of the statistical uncertainty of each edge RBFE. A scatter plot comparing relative vs calculated binding free energies is produced, this is only possible because experimental data is available for this dataset.

Next, the set of RBFEs is converted into a set of binding free energies with an arbitrary reference value using the *cinnabar* software to produce a scatter plot of calculated vs experimental binding free energies, together with statistical measures of accuracy (mean unsigned error, root mean squared error, Pearson coefficient, and Kendall tau coefficient). The functionality covered by this notebook is illustrated in Figure 9.

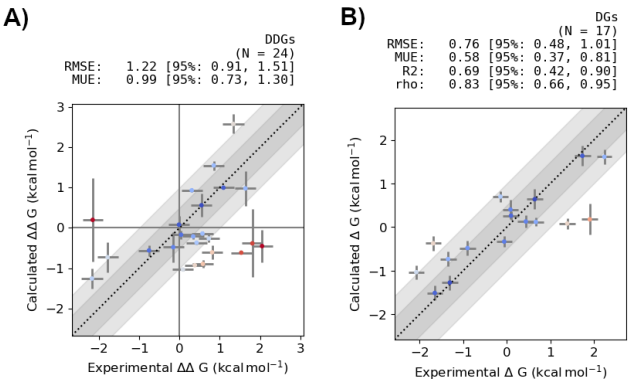

Figure 9. A) Scatter plot of experimental vs computed pairwise $\Delta\Delta G_{bind}$ free energies from the analysed perturbation network. **B)** Scatter plot of ΔG_{bind} estimates after processing of the network.

3.4.4 ABFE calculations

The fourth notebook 01-setup-abfe describes how to set up alchemical ABFE calculations with BSS for use with either *mdrun* or *SOMD*. This free energy functionality is currently implemented in Exscientia's sandpit area of this version of BioSimSpace. Sandpits are used to test experimental features without accidentally breaking core functionality of the toolkit.

Although RBFE calculations can be very useful in drug discovery, several important problems lie outside the scope of standard RBFE calculations. These include the calculation of the binding free energies of structurally dissimilar ligands to a common target; the calculation of the binding free energies of the same ligand to the same protein with different binding poses; the calculation of the binding free energies of a single ligand to a range of targets, as would be required to optimize selectivity or promiscuity. These quantities can be calculated using alchemical ABFE calculations because they utilize a more general *double decoupling* thermodynamic cycle than in RBFE calculations [33]. This involves entirely removing the ligand's intermolecular interactions in the presence of restraints between the ligand and receptor, as shown in Figure 10A.

The notebook describes the use of *BSS.Align.decouple* to tag a molecule in a BSS system that should have all its intermolecular interactions removed to compute its absolute binding free energy. This is illustrated here with the use of the ligand MIF180 bound to the protein MIF [34, 35].

Next, it is shown how to run and analyse a simulation of the fully-interacting protein-ligand complex to generate the required intermolecular restraints. Both actions are handled using *BSS.FreeEnergy.RestraintSearch*. The popular 6-degrees of freedom Boresch restraints are supported [36], as well as a Clark's multiple distance restraints (MDR)

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

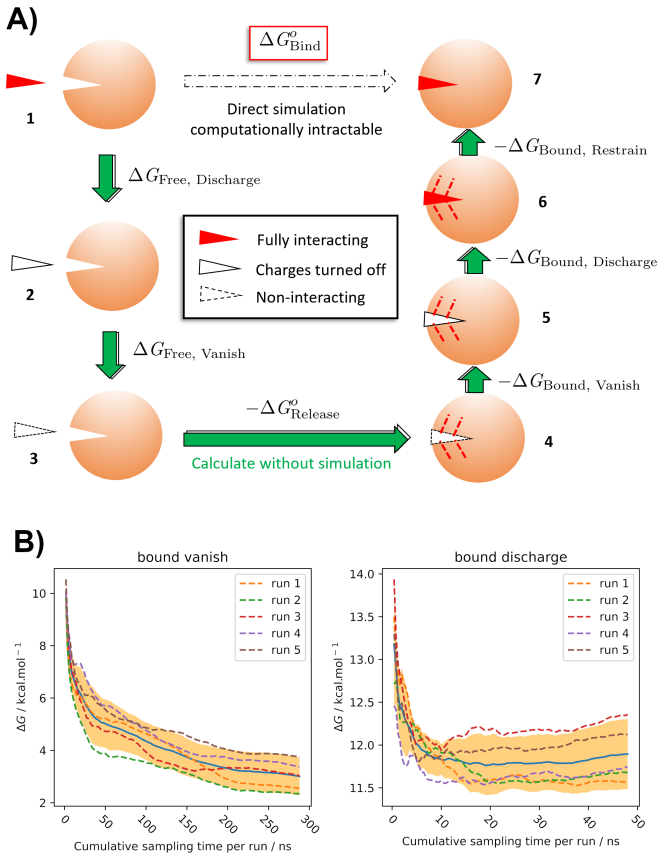

Figure 10. **A)** Thermodynamic cycle for alchemical absolute binding free energy calculations. **B)** Sample convergence plots for legs 5 and 6 of the ABFE thermodynamic cycle for the MIF/MIF180 complex, showing especially poor convergence for the bound vanish stage prior to discarding initial non-equilibrated samples.

methodology [35]. The notebook illustrates the automated generation of Boresch or MDR restraints with BSS. The algorithm implemented aims to select stable restraints which mimic strong native receptor-ligand interactions. The notebook includes visualization of the chosen restraints before free energy inputs are prepared, which allows the user to identify cases where the selected restraints may not be optimal, or where symmetry corrections may be required [37]. The notebook then demonstrates the use of *BSS.FreeEnergy.AlchemicalFreeEnergy* to generate input files for the engines SOMD or GROMACS. This class in the Exscientia sandpit contains all of the functionality from *BSS.FreeEnergy.Relative* in the main version of the code, as well as additional functionality required for ABFE calculations.

Since ABFE calculations can be time-consuming we recommend parallel execution of the different legs of the thermodynamic cycle using an approach similar to that used in the RBFE tutorial.

The fifth notebook 02-analysis-abfe describes how to analyse an ABFE calculation to estimate the free energy of binding of a ligand. Sample output simulation data are provided for each leg of the double decoupling thermodynamic cycle and analysed using *BSS.FreeEnergy.AlchemicalFreeEnergy.Analyse* to plot potentials of mean forces. The standard free energy of binding is then obtained by summing the free energy changes from each leg and adding a standard state correction term for the use of Boresch restraints, along with any symmetry corrections required. The notebook also describes how to carry out convergence analyses (see Figure 10B) to assess the robustness of the ABFE estimates.

4 Author Contributions

LH prepared Tutorial 1, DL and LH prepared Tutorial 2, AH and LH prepared Tutorial 3, JS, LH, AH, FC, and JM prepared Tutorial 4, which was built on older tutorial material by AM and SB and on contributions from ZW, MS, and BC. MB and CW reviewed and tested all tutorials and ported the tutorials to a web server. The authors are listed in alphabetical order, with the exception of the first coauthor. For a more detailed description of author's contributions, see the GitHub issue tracking and changelog at https://github.com/OpenBioSim/biosimspace_tutorials.

5 Other Contributions

Gratitude is expressed to the users of BioSimSpace who have given important feedback over the past years that have influenced the production of our tu-

https://doi.org/10.33011/livecoms.5.1.2375
Living J. Comp. Mol. Sci. 2023, 5(1), 2375

torials and documentation. For a more detailed description of contributions from the community and others, see the GitHub issue tracking and changelog at https://github.com/OpenBioSim/biosimspace_tutorials.

6 Potentially Conflicting Interests

JM is a member of the Scientific Advisory Board of Cresset.

7 Funding Information

JM acknowledges support from an EPSRC standard grant (EP/P022138/1) and from the University of Edinburgh UCB via an EPSRC-Impact Acceleration Account (IAA PIII074).

Author Information

ORCID:

Lester Hedges: 0000-0002-5624-0500
Matthew Burman: 0000-0001-5360-0024
Sofia Bariami: 0000-0002-5240-7684
Finlay Clark: 0000-0003-0474-5475
Benjamin P. Cossins: 0000-0002-6699-8833
Adele Hardie: 0009-0002-9943-9920
Anna M. Herz: 0000-0003-2831-6691
Dominykas Lukauskis: 0000-0002-4999-2691
Antonia Mey: 0000-0001-7512-5252
Julien Michel: 0000-0003-0360-1760
Jenke Scheen: 0000-0001-9781-0445
Miroslav Suruzhon: 0000-0002-6794-1679
Christopher J Woods: 0000-0001-6563-9903
Zhiyi Wu: 0000-0002-7615-7851

References

[1] **Hedges LO**, Mey ASJS, Laughton CA, Gervasio FL, Mulholland AJ, Woods CJ, Michel J. BioSimSpace: An interoperable Python framework for biomolecular simulation. Journal of Open Source Software. 2019; 4(43):1831. doi: 10.21105/joss.01831.

[2] **Case DA**, Belfon K, Ben-Shalom IY, Brozell SR, Cerutti DS, III TEC, Cruzeiro VWD, Darden TA, Duke RE, Giambasu G, Gilson MK, Gohlke H, Goetz AW, Harris R, Izadi S, Izmailov SA, Kasavajhala K, Kovalenko A, Krasny R, Kurtzman T, et al.; 2020, aMBER 2020.

[3] **Abraham MJ**, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX. 2015; 1:19–25. doi: 10.1016/j.softx.2015.06.001.

[4] **Phillips JC**, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kale L, Schulten K. Scalable molecular dynamics with NAMD. Journal of Computational Chemistry. 2005; 26(16):1781–1802. doi: 10.1002/jcc.20289.

[5] **Eastman P**, Swails J, Chodera JD, McGibbon RT, Zhao Y, Beauchamp KA, Wang LP, Simmonett AC, Harrigan MP, Stern CD, Wiewiora RP, Brooks BR, Pande VS. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. PLOS Computational Biology. 2017; 13(7):e1005659. doi: 10.1371/journal.pcbi.1005659.

[6] **Richard J Gowers**, Max Linke, Jonathan Barnoud, Tyler J E Reddy, Manuel N Melo, Sean L Seyler, Jan Domanski, David L Dotson, Sebastien Buchoux, Ian M Kenney, Oliver Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In: Sebastian Benthall, Scott Rostrup, editors. *Proceedings of the 15th Python in Science Conference*; 2016. p. 98 – 105. doi: 10.25080/Majora-629e541a-00e.

[7] **McGibbon RT**, Beauchamp KA, Harrigan MP, Klein C, Swails JM, Hernández CX, Schwantes CR, Wang LP, Lane TJ, Pande VS. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. Biophysical Journal. 2015; 109(8):1528 – 1532. doi: 10.1016/j.bpj.2015.08.015.

[8] **Beckstein O**, Dotson D, Wu Z, Wille D, Marson D, Kenney I, shuail, Lee H, trje3733, Lim V, Schlaich A, Alibay I, Hénin J, Barhaghi MS, Merz P, Joseph T, Hsu WT, alchemistry/alchemlyb: 2.0.1. Zenodo; 2023. doi: 10.5281/zenodo.7809117.

[9] **Huggins DJ**, Biggin PC, Dämgen MA, Essex JW, Harris SA, Henchman RH, Khalid S, Kuzmanic A, Laughton CA, Michel J, Mulholland AJ, Rosta E, Sansom MSP, van der Kamp MW. Biomolecular simulations: From dynamics and mechanisms to computational assays of biological activity. WIREs Computational Molecular Science. 2018; 9(3). doi: 10.1002/wcms.1393.

[10] **Braun E**, Gilmer J, Mayes HB, Mobley DL, Monroe JI, Prasad S, Zuckerman DM. Best Practices for Foundations in Molecular Simulations [Article v1.0]. Living Journal of Computational Molecular Science. 2019; 1(1). doi: 10.33011/livecoms.1.1.5957.

[11] **Limongelli V**, Bonomi M, Parrinello M. Funnel metadynamics as accurate binding free-energy method. Proceedings of the National Academy of Sciences. 2013; 110(16):6358–6363. doi: 10.1073/pnas.1303186110.

[12] **Isralewitz B**, Gao M, Schulten K. Steered Molecular Dynamics and Mechanical Functions of Proteins. Curr Opin Struct Biol. 2001; 11:224–230. doi: 10.1016/s0959-440x(00)00194-9.

[13] **Bussi G**, Tribello GA. Analyzing and Biasing Simulations with PLUMED. Bonomi M, Camilloni C, editors, New York, USA: Springer Science; 2019. doi: 10.1007/978-1-4939-9608-7_21.

[14] **Mey ASJS**, Allen BK, Macdonals HEB, Chodera JD, Hahn DF, Kuhn M, Michel J, Mobley DL, Naden lN, Prasad S, Rizzi A, Scheen J, Shirts MR, Tresadern G, Xu H. Best Practices for Alchemical Free Energy Calculations [Article v1.0]. Living Journal of Computational Molecular Science. 2020; 2(1):18378. doi: 10.33011/livecoms.2.1.18378.

[15] **Evans R**, Hovan L, Tribello GA, Cossins BP, Estarellas C, Gervasio FL. Combining Machine Learning and Enhanced Sampling Techniques for Efficient and Accurate Calculation of Absolute Binding Free Energies. Journal of Chemical Theory and Computation. 2020; 16(7):4641–4654. doi: 10.1021/acs.jctc.0c00075.

[16] **Saleh N**, Ibrahim P, Saladino G, Gervasio FL, Clark T. An Efficient Metadynamics-Based Protocol To Model the Binding Affinity and the Transition State Ensemble of G-Protein-Coupled Receptor Ligands. Journal of Chemical Information and Modeling. 2017; 57(5):1210–1217. doi: 10.1021/acs.jcim.6b00772.

[17] **Zwier MC**, Chong LT. Reaching biological timescales with all-atom molecular dynamics simulations. Current Opinion in Pharmacology. 2010; 10(6):745–752. doi: https://doi.org/10.1016/j.coph.2010.09.008.

[18] **Choy MS**, Li Y, Machado LESF, Kunze MBA, Connors CR, Wei X, Lindorff-Larsen K, Page R, Peti W. Conformational Rigidity and Protein Dynamics at Distinct Timescales Regulate PTP1B Activity and Allostery. Mol Cell. 2017; 65:644–658. doi: 10.1016/j.molcel.2017.01.014.

[19] **Wells DB**, Abramkina V, Aksimentiev A. Exploring transmembrane transport through -hemolysin with grid-steered molecular dynamics. The Journal of Chemical Physics. 2007; 127. doi: 10.1063/1.2770738, 125101.

[20] JADE2 benchmark;. https://www.hecbiosim.ac.uk/access-hpc/our-benchmark-results/jade2-benchmarks.

[21] **Tribello GA**, Bonomi M, Branduardi D, Camilloni C, Bussi G. PLUMED 2: New Feathers for an Old Bird. Comput Phys Commun. 2014; 185:604–613. doi: 10.1016/j.cpc.2013.09.018.

[22] **Ahmad F**, Li PM, Meyerovitch J, Goldstein BJ. Osmotic loading of neutralizing antibodies demonstrates a role for protein-tyrosine phosphatase 1B in negative regulation of the insulin action pathway. Journal of Biological Chemistry. 1995; 270:20503–20508. https://doi:10.1074/jbc.270.35.20503.

[23] **Wiesmann C**, Barr1 KJ, Kung J, Zhu J, Erlanson DA, Shen W, Fahr BJ, Zhong M, Taylor L, Randal1 M, McDowell1 RS, Hansen SK. Allosteric Inhibition of Protein Tyrosine Phosphatase 1B. Nature Structural and Molecular Biology. 2004; 11:730–737. doi: 10.1038/nsmb803.

[24] **Potterton A**, Husseini FS, Southey MWY, Bodkin MJ, Heifetz A, Coveney PV, Townsend-Nicholson A. Ensemble-Based Steered Molecular Dynamics Predicts Relative Residence Time of A2A Receptor Binders. Journal of Chemical Theory and Computation. 2019; 15(5):3316–3330. doi: 10.1021/acs.jctc.8b01270.

[25] **Wehmeyer C**, Scherer MK, Hempel T, Husic BE, Olsson S, Noé F. Introduction to Markov state modeling with the PyEMMA software [Article v1.0]. Living Journal of Computational Molecular Science. 2019; 1(1):5965. doi: 10.33011/livecoms.1.1.5965.

[26] **Hardie A**, Cossins BP, Lovera S, Michel J. Deconstructing allostery by computational assessment of the binding determinants of allosteric PTP1B modulators. Communications Chemistry. 2023; 6(1). doi: 10.1038/s42004-023-00926-1.

[27] **Cournia Z**, Allen B, Sherman W. Relative Binding Free Energy Calculations in Drug Discovery: Recent Advances and Practical Considerations. Journal of Chemical Information and Modeling. 2017; 57(12):2911–2937. doi: 10.1021/acs.jcim.7b00564.

[28] **Kuhn M**, Firth-Clark S, Tosco P, Mey ASJS, Mackey M, Michel J. Assessment of Binding Affinity via Alchemical Free-Energy Calculations. Journal of Chemical Information and Modeling. 2020; 60(6):3120–3130. doi: 10.1021/acs.jcim.0c00165.

[29] **Hahn DF**, Bayly CI, Boby ML, Macdonald HEB, Chodera JD, Gapsys V, Mey ASJS, Mobley DL, Benito LP, Schindler CEM, Tresadern G, Warren GL. Best Practices for Constructing, Preparing, and Evaluating Protein-Ligand Binding Affinity Benchmarks [Article v1.0]. Living Journal of Computational Molecular Science. 2022; 4(1). doi: 10.33011/livecoms.4.1.1497.

[30] **Calabrò G**, Woods CJ, Powlesland F, Mey ASJS, Mulholland AJ, Michel J. Elucidation of Nonadditive Effects in Protein–Ligand Binding Energies: Thrombin as a Case Study. The Journal of Physical Chemistry B. 2016; 120(24):5340–5350. doi: 10.1021/acs.jpcb.6b03296.

[31] **Wang L**, Wu Y, Deng Y, Kim B, Pierce L, Krilov G, Lupyan D, Robinson S, Dahlgren MK, Greenwood J, Romero DL, Masse C, Knight JL, Steinbrecher T, Beuming T, Damm W, Harder E, Sherman W, Brewer M, Wester R, et al. Accurate and Reliable Prediction of Relative Ligand Binding Potency in Prospective Drug Discovery by Way of a Modern Free-Energy Calculation Protocol and Force Field. Journal of the American Chemical Society. 2015; 137(7):2695–2703. doi: 10.1021/ja512751q.

[32] **Liu S**, Wu Y, Lin T, Abel R, Redmann JP, Summa CM, Jaber VR, Lim NM, Mobley DL. Lead optimization mapper: automating free energy calculations for lead optimization. Journal of Computer-Aided Molecular Design. 2013; 27(9):755–770. doi: 10.1007/s10822-013-9678-y.

[33] **Gilson MK**, Given JA, Bush BL, McCammon JA. The Statistical-Thermodynamic Basis for Computation of Binding Affinities: A Critical Review. Biophys J. 1997; 72(3):1047–1069. doi: 10.1016/S0006-3495(97)78756-3.

[34] **Qian Y**, de Vaca IC, Vilseck JZ, Cole DJ, Tirado-Rives J, Jorgensen WL. Absolute Free Energy of Binding Calculations for Macrophage Migration Inhibitory Factor in Complex with a Druglike Inhibitor. The Journal of Physical Chemistry B. 2019; 123(41):8675–8685. doi: 10.1021/acs.jpcb.9b07588.

[35] **Clark F**, Robb G, Cole DJ, Michel J. Comparison of Receptor–Ligand Restraint Schemes for Alchemical Absolute Binding Free Energy Calculations. Journal of Chemical Theory and Computation. 2023; doi: 10.1021/acs.jctc.3c00139.

[36] **Boresch S**, Tettinger F, Leitgeb M, Karplus M. Absolute Binding Free Energies: A Quantitative Approach for Their Calculation. J Phys Chem B. 2003; 107(35):9535–9551. doi: 10.1021/jp0217839.

[37] **Duboué-Dijon E**, Hénin J. Building Intuition for Binding Free Energy Calculations: Bound State Definition, Restraints, and Symmetry. J Chem Phys. 2021; 154(20):204101. doi: 10.1063/5.0046853.