Dimorphite-DL and Biotite-tools, Two Open Source Programs for the Acceleration of Structure-based Drug Design

by

Jesse Cianan Kaminsky

Submitted to the Graduate Faculty of

University Honors College in partial fulfillment

of the requirements for the degree of

Bachelor of Philosophy

University of Pittsburgh

2019

UNIVERSITY OF PITTSBURGH

UNIVERSITY HONORS COLLEGE

This thesis/dissertation was presented

by

Jesse Cianan Kaminsky

It was defended on

March 28, 2019

and approved by

Alemayehu Gorfe, Associate Professor, UTHealth McGovern Medical School

Andrea Berman, Assistant Professor, Department of Biological Sciences

Lillian Chong, Associate Professor, Department of Chemistry

Thesis Advisor/Dissertation Director: Jacob Durrant, Assistant Professor, Department of Biological Sciences

Dimorphite-DL and Biotite-tools, Two Open Source Programs for the Acceleration of Structure-based Drug Design

Jesse Cianan Kaminsky, BPhil

University of Pittsburgh, 2019

Computer-aided drug design has seen a proliferation of tools that allow the manipulation of small molecule and macromolecular structures in increasingly high-throughput settings. Molecular dynamics simulations, small molecule docking software, and visualization tools allow researchers to rapidly identify drug candidates and narrow the list of compounds that experimentalists must consider for further testing. Any gap in automating computer-aided drug design thus delays potentially life-saving discoveries. Here we present two open-source programs we developed to address challenges facing both protein and ligand preparation. **Dimorphite-DL** is a lightweight python program that predicts protonation states of small molecules using an empirical approach to ensure accurate docking and modelling calculations. The presence or absence of a hydrogen atom often determines whether a given ligand will bind a protein of interest. **Biotite-tools** is a python package that provides several popular statistical functions for analyzing molecular dynamics simulations in an easy-to-use way. Conformational fluctuation is complex, and it can be challenging to extract insight from what is essentially a "protein movie." As such, simulation analysis has largely been restricted to those with backgrounds in computation, limiting the scope of such a powerful tool. Biotite-tools aims to accelerate the efforts of those already working with molecular dynamics and make analysis more accessible to experimentalists.

Table of Contents

List of Tables

List of Figures

Preface

I would like to thank the members of Dr. Jacob Durrant's lab for all their valuable insight and contributions to dimorphite-DL and biotite-tools. I would like to thank Patrick Ropp for his extensive guidance and mentorship regarding software development and Python, as well as his leadership and initial development of the dimorphite-DL project. Thank you to Jacob Spiegel for additional insight into implementation details and helpful code discussions. Thank you to James Haddad for your assistance in the development process. Thank you to Kevin Cassidy and Dr. Andrea Berman's lab for their adoption of the biotite-tools package in researching the LARP1 DM15 region. Thank you to the Center for Research Computing and the American Cancer Society for the use of their resources and funding to complete the LARP1 DM15 MD simulations. Thank you to the Department of Biological Sciences for the opportunity to present my research at the Science 2018 Undergraduate Poster Session. Finally, thank you Dr. Durrant for the mentorship, support, and great advice you have provided me in pursuing my first major research experience.

1.0 Introduction

1.1 **Narrowing chemical space in drug discovery**

Structure-based drug design is the pursuit of pharmaceutically relevant compounds that directly act on specific proteins. There are many challenges to this approach. Identifying the protein target for treatment involves years of experimental investigation. Resolving the structure of such a protein, a critical first step in structure-based drug design, is a complicated process in itself. Once a structure is known, we are interested in identifying ligands that might inhibit or enhance functionality. How do we select which molecules to investigate as potential ligands? Even after reducing chemical space by common drug requirements (membrane permeability, stability, ease of synthesis, free of off-target effects, etc…), considering every possible compound is still an intractable task (1).

Virtual screening (VS) allows us to select from a smaller pool of molecules in seeking novel disease treatments (2). Given a static structure derived from the crystallized protein or family nuclear magnetic resonance (NMR) structures, VS identifies ligand candidates from extensive lists of input molecules. The ability to handle large quantities of such data in an efficient and iterative manner is termed “high-throughput.” A high-throughput approach to VS quickly generates pools of drug candidates for further experimental study. VS uses docking software to accept only those molecules with potentially significant binding capability.

Docking software assigns scores that reflect the binding affinity of a given compound to a protein of interest (3). It is vital to consider only biologically relevant states in such a process (4, 5). Molecules move between many conformations and states. Docking accounts for ligand motion by generating different scores depending on how the compound is fit into the binding site, but it does not account for every possible ionization state of the ligand. VS uses molecules precisely as specified by the input.

1.2 **The importance of ionization states in virtual screens**

The protonation state of a ligand can make a major difference in binding affinity (6). Consider an active site containing a prominent aspartic acid residue at physiological pH. Its negative charge will bind a protonated ligand presenting a positive charge with far more stability than the deprotonated form. There is thus a need to ensure only the properly protonated state at a given pH is docked onto the protein. While many programs serve this need, most are too slow for high-throughput VS, too expensive, or too inaccurate (7). Dimorphite-DL seeks to fill this functional gap and has been published in the *Journal of Cheminformatics*. Without such a program, VS must consider excessive or incomplete datasets of protonation states. Pre-existing tools that rely on quantum-mechanical calculations force the throughput of VS to be rate dependent on pK_a prediction.

1.3 **Sampling ensembles of conformations with molecular dynamics simulations**

While computational resources limit the exhaustive screening of ligand states, protein states are instead limited by physiological relevance. Proteins exist in complex environments surrounded by water, ions, and other macromolecules. Proteins in such environments possess kinetic and chemical potential energies that are not accurately captured by the necessary experimental conditions involved in structure elucidation. Protein structures determined by x-ray crystallography or NMR do not fully account for conformational diversity *in vivo* (8). Conformations and dynamic states should only be prioritized by their relative energetics. Docking a ligand onto a crystal structure is comparable to fitting a glove onto a fist.

Molecular dynamics (MD) simulations are a prominent solution to sampling more physiologically relevant conformations (8). These simulations rely on mathematical descriptions of potential energies to calculate the forces between atoms in space, given specific atom types, distances, and bonding (9). Applying such forces to sets of atomic coordinates models a starting structure in a highly customizable context, such as physiological conditions. By recording the change in the atomic coordinates over time, an entire **trajectory** of conformational states emerges (10). Each moment of time is represented as an individual "frame" of the trajectory. Docking then calculates binding affinities for all prominent conformations, ensuring a pool of *more* physiologically relevant ligands.

Many problems arise in performing MD simulations. The algorithms rely on simplifications to minimize the computational cost of such intensive calculations (11). We often record the positions of every single atom being simulated (protein and solvent) on picosecond timescales, and microsecond length simulations have become routine. Thus not only do

simulations take a very long time to run, even on state-of-the-art parallel processor systems (8), but interpreting the data becomes a challenge. How do we begin to analyze a protein in motion?

There are numerous open-source libraries, mostly written in Python or R, that provide statistical analysis functions and frameworks for understanding MD simulations (12). These libraries, while effective and widely used among the MD community, generally do not directly integrate with other plotting and visualization libraries like Matplotlib (13) or Pandas (14). Some, like MDAnalysis (15), provide tutorials to assist researchers in understanding the outputs of their functions. Biotite-tools provides pre-constructed scripts that link analysis with visualization, with built-in customizability that requires no prior knowledge of Python or coding. Such scripts could be wrapped in an easy-to-use graphical interface and executed on servers, enabling those with little computational expertise to perform complex statistical analyses on their MD data. The eventual goal is for anyone studying structural biology to perform a MD simulation, *easily* understand its implications, identify ligands, and share the results, even with limited access to computational resources.

1.4 **Sustainable and efficient biomedical tool development**

Excepting the open-source biomedical community, much of today's state of the art software is only accessible behind paywalls. Restrictive licensing also prevents open source use of pre-existing codebases in developing further applications. It is important to prioritize modularity of biomedical research tools to allow incorporation into future projects (such as a comprehensive server for structure-based drug design). Even within open-source development, code often suffers from unclear documentation or a failure to generalize to new contexts. Consequentially, software

that performs admirably well at its individual purpose is often excluded from further development. The only solution is ensuring all published code is well documented, reasonably maintained, and constructed with an eye towards modularity. We developed dimorphite-DL and biotite-tools to be as accessible as possible, in hopes of encouraging others to use our code and incorporate it into their own research and tool development.

2.0 Materials and Methods

2.1 **Dimorphite-DL Implementation**

Dimorphite-DL (16) works by identifying known ionizable structures within input small molecules and predicting their states at a given pH range based on a collection of pre-existing pK_a values associated with each of 38 different moieties. We calculated these values from 1,938 molecules containing individual sites with experimentally determined pK_as. We also used 78 instances of phosphates and phosphonates that we treated differently as they can be doubly protonated. We compiled these molecules from a publication by Lee *et al*. and from public, online databases (17, 18). We found the mean pK_a and standard deviation across all compounds containing each moiety. The structure of these moieties and the pK_a dimorphite-DL associates with each can be found in Figure 1.

To more accurately account for the diversity of pK_a values of moieties, we chose to use a pK_a range instead of a single value. We construct this range using the mean pK_a value (μ) and standard deviation (σ) of an identified moiety, scaled by a user parameter we call the pK_a precision factor (n). The range is defined as $[\mu - n\sigma, \mu + n\sigma]$. Dimorphite-DL compares this range to the provided pH range. For a pK_a range entirely below the pH range, the moiety is deprotonated. For a pK_a range entirely above the pH range, the moiety is protonated. For ranges with any overlap, both forms are generated. Figure 2 visually depicts this process.

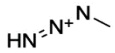
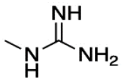
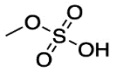
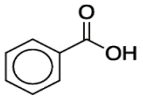
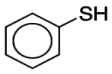
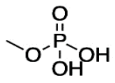
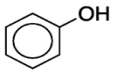
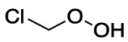
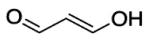
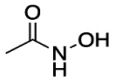
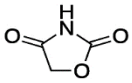
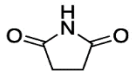
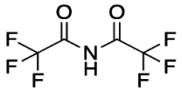
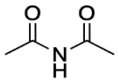
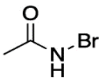
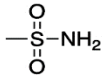
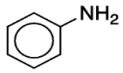
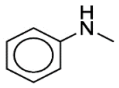
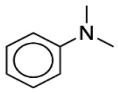
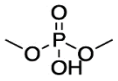
Azide  4.65 ± 0.07	**Nitro**  -1000 ± 0	**AmidineGuanidine1**  12.03 ± 1.59	**AmidineGuanidine2**  10.04 ± 2.13	**Sulfate**  -2.36 ± 1.30
Sulfonate  -1.82 ± 1.41	**Sulfinic Acid**  1.79 ± 0.44	**Phenyl Carboxyl**  3.46 ± 1.25	**Carboxyl**  3.46 ± 1.29	**Thioic Acid**  0.68 ± 1.50
Phenyl Thiol  4.98 ± 2.61	**Thiol** 9.12 ± 1.33	**Phosphate**  2.42 ± 1.11 6.51 ± 0.95	**Phosphonate**  1.88 ± 7.25 0.59 ± 0.85	**Phenol**  7.07 ± 3.28
Peroxide1  8.74 ± 0.76	**Peroxide2**  11.98 ± 0.87	**O=C-C=C-OH**  3.55 ± 0.80	**Vinyl Alcohol**  8.87 ± 1.66	**Alcohol**  14.78 ± 2.55
N-hydroxyamide  9.30 ± 1.22	**Ringed Imide1**  6.45 ± 0.56	**Ringed Imide2**  8.68 ± 1.87	**Imide**  2.47 ± 1.48	**Imide2**  10.23 ± 1.12
Amide Electronegative  3.49 ± 2.69	**Amide**  12.01 ± 4.51	**Sulfonamide**  7.92 ± 1.98	**Anilines Primary**  3.90 ± 2.07	**Anilines Secondary**  4.34 ± 2.18
Anilines Tertiary  4.17 ± 2.01	**Aromatic Nitrogen Unprotonated**  4.35 ± 2.07	**Amines Primary Secondary Tertiary**  8.16 ± 2.52	**Phosphinic Acid**  2.97 ± 0.69	**Phosphate Diester**  2.73 ± 2.54
Phosphonate Ester  2.09 ± 0.45	**Primary Hydroxyl Amine**  4.04 ± 0.85	**Aromatic Nitrogen Protonated**  12.08 ± 5.10		

Figure 1 Dimorphite-DL's 38 supported moiety structures

The mean pK_a and standard deviation calculated from each respective set of experimentally determined pK_a values are shown. Nitro groups are assigned an arbitrary pK_a of -1000 to ensure the moiety is always deprotonated (16).

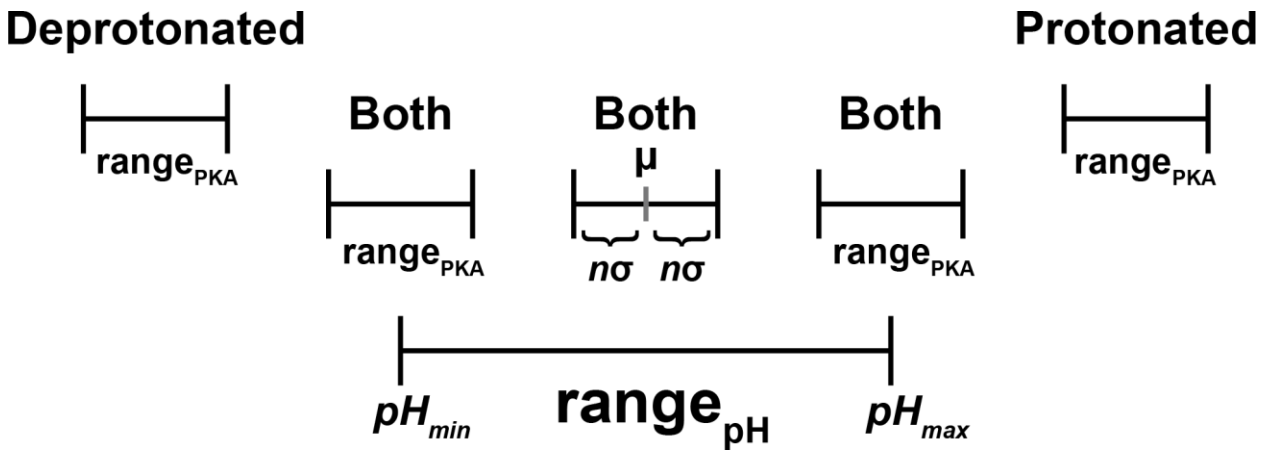


Figure 2 Dimorphite-DL algorithm schematic
Dimorphite-DL deprotonates any structure that presents a moiety with a pK_a range entirely below the user-provided pH range and protonates any entirely above (16). For any pK_a range that overlaps with the pH range, both structures are generated. pK_a ranges are calculated from a trained mean pK_a, scaled by the standard deviation times the precision factor.

Dimorphite-DL uses the SMILES syntax for receiving input molecules from the user. The SMILES language provides a way of representing molecular structures as 1-dimensional text (19). Given a set of SMILES or a file containing SMILES, dimorphite-DL uses the open-source Python library RDKit (20) to search through each molecule for any structures that match one of the supported 38 moieties, and outputs the correctly protonated SMILES. For *each* ionizable site found, protonation is assigned independently, producing as many output structures as necessary for the given pH range.

Dimorphite-DL functions on macOS High Sierra, Ubuntu 18.04, and Windows 10 and requires the user to have installed Python 2.7 or 3.6 or higher, as well as RDKit 2016.09.2 or higher. Source code, installation instructions, and a brief guide can be found at http://durrantlab.com/dimorphite-dl/. A basic testing suite is also provided to demonstrate the functionality of individual components of dimorphite-DL.

2.2 **Evaluating Dimorphite-DL Accuracy**

Because dimorphite-DL relies on experimentally determined pK_a values, the limited data that might be used to evaluate its accuracy had already been used to calculate each moiety's pK_a value. Although the set of training data was only used to compute a linear model for each moiety, it would be problematic to test the accuracy of such models on the same data that produced them. For this reason, we chose to use 3-fold cross validation to evaluate the accuracy of our method. The training set was broken into thirds. We trained Dimorphite-DL on each combination (fold) of two of these thirds, before testing it on the final third. The resulting values were averaged to obtain final accuracies for *each* moiety independently. Note that any displayed standard deviations reflect the standard deviation across folds, *not* across moieties.

In evaluating the accuracy of our algorithm, we broke up its output into three distinct categories. A **correct** prediction indicates that *only* the correct state of a moiety in a given pH range was generated, be it protonated, deprotonated, or both. An **excess** prediction indicates that *at least one* incorrect structure was generated in addition to the correct one. An **incorrect** prediction indicates that at least one *correct* state was *not* generated. While an excess prediction would simply add unnecessary molecules to VS, an incorrect prediction would render further analysis fundamentally irrelevant. For all outcomes, we used the experimentally derived pK_a value of the SMILE to determine the accuracy of the resulting structure at the given pH range. We measured the outcomes as the percentage of the dataset they comprised. These outcomes are defined as follows (16):

1. Dimorphite-DL predicts the correct state
    a. $pK_a < pH_{min}$, and dimorphite-DL deprotonates the compound
    b. $pK_a > pH_{max}$, and dimorphite-DL protonates the compound
    c. $pH_{min} \leq pK_a \leq pH_{max}$, and dimorphite-DL generates both deprotonated and protonated forms

2. Dimorphite-DL predicts an excess state (i.e., two states when only one is appropriate)
    a. $pK_a < pH_{min}$ or $pK_a > pH_{max}$, but dimorphite-DL generates both deprotonated and protonated forms
3. Dimorphite-DL predicts the incorrect (or incomplete) state
    a. $pK_a < pH_{min}$, but dimorphite-DL protonates the compound
    b. $pK_a > pH_{max}$, but dimorphite-DL deprotonates the compound
    c. $pH_{min} \leq pK_a \leq pH_{max}$, and dimorphite-DL either deprotonates or protonates the compound (not both)

Because phosphates and phosphonates can be protonated twice, they were evaluated according to a more complicated set of rules that nevertheless follow the same line of reasoning. We did not evaluate the accuracy of nitro groups due to their minimal presence within the training data. The moiety accuracies were evaluated at default pH and pK_a precision factor values (physiological pH 6.4-8.4 and precision factor 1.0). Dimorphite-DL considers sites independently even on molecules containing more than one, so this evaluation on single-site structures extrapolates to multi-site structures as well.

To evaluate the effect of the precision factor, we considered accuracy across the entire training set (not using 3-fold cross validation). An average was calculated across all moieties weighted by the respective proportion of each within the dataset.

2.3 **Biotite-tools Implementation**

Much of the biotite-tools original codebase lies in the infrastructure through which it handles user input. Because biotite-tools is a collection of scripts, it inherently lacks much of the adaptability that Python libraries provide. In exchange for this limitation, users are not required to understand the details of the computations being performed. Users must only know the implications of each analysis and what can be accomplished with the results of each. That said, a

complete loss of versatility in these functions would render biotite-tools obsolete. Many of our implementation choices reflect a need to find a balance between effectiveness and ease-of-use.

Biotite-tools uses the open-source Python library MDAnalysis (15) to perform the bulk of its statistical analyses. Binary MD trajectory files are also interpreted in memory using MDAnalysis' parsing modules. After applying MDAnalysis, biotite-tools uses the common data manipulation and plotting packages NumPy (21) and matplotlib (13) to perform further analysis and provide output to the user. Matplotlib enables biotite-tools to create publication quality figures as scalable vector graphics files, allowing further modification in the user's plotting software of choice.

Biotite-tools uses Python's argparse module to handle user input. While a complete description of every parameter is beyond the scope of this thesis, it can be found in biotite-tools' documentation, together with helpful examples and descriptions. Using biotite-tools is as simple as running any of the scripts with Python and providing any parameter desired. All parameters can also be provided as a json file to enhance accessibility by a server environment. Because each script performs different computations, each possesses its own selection of parameters that fine tune their functionality. However, some parameters generalize to every script. Table 1 provides a brief overview of the more important parameters users can pass to *any* script.

Biotite-tools functions on macOS High Sierra, Ubuntu 18.04, and Windows 10 and requires the user to have installed Python 2.7 or 3.6 or higher, as well as MDAnalysis, NumPy, matplotlib, and optionally scipy (22) or scikit-learn (23). Source code, installation instructions, and extensive documentation can be found at https://git.durrantlab.pitt.edu/jdurrant/biotite-mdanalysis.

Table 1 Summary of the more significant universal biotite-tools user parameters

User Parameter	Function
--top_file & --coor_file	These parameters specify the input data files for biotite-tools to analyze.
--dir	Tells biotite-tools to search the current directory for all compatible files. Multiple trajectories can be handled.
--compare	When multiple trajectories are provided, they can either be analyzed in batch, or if --compare is present, directly compared to one another.
--output_dir	What directory to save any output files.
--selection	What portion of the simulated system to consider. Users can choose to only consider certain groups of atoms (for example only alpha carbons).
--on_server	Allows biotite-tools to be run over an SSH client without throwing matplotlib errors.

2.3.1 **Aligning Trajectories**

Biotite-tools uses MDAnalysis' (15) aligning module to read in a trajectory, align it according to a specified atom selection, and write it in the same file format as provided. Most of the other scripts also perform alignment prior to analysis and as such this atom selection can be provided to any script in use.

2.3.2 **Pruning and Trimming Trajectories**

Pruning and trimming is performed in memory using MDAnalysis (15). Users provide the percentage of a simulation they wish to keep from the end of a given simulation. Users also provide the maximum number of frames they wish the output trajectory to contain. Input trajectories are

then cut and every N frames are selected to meet these specifications. Trajectories are saved in the same file format provided. To allow a server to easily access a trajectory for visualization in-browser, frames can also each be saved as a PDB file.

2.3.3 **Root Mean Square Deviation (RMSD)**

RMSD values are calculated by MDAnalysis (15) as the square root of the sum of the square distances between atoms at a given frame and the first frame of the trajectory (24). These values are then plotted against the timestep (users must specify the amount of time between frames used by their respective MD engine), along with a running average calculated with NumPy (21). Users can specify the lengths of the x (time) and y (RMSD value) axis to enable multi-trajectory comparison. Users can also choose to directly overlay up to three trajectories on one plot. Trajectories are aligned prior to analysis.

2.3.4 **Root Mean Square Fluctuation (RMSF)**

RMSF values are calculated for atom selections by MDAnalysis as the square root of the sum of the square distances for a given atom across the whole trajectory. Using an original implementation of the same algorithm (25) MDAnalysis uses, biotite-tools offers the same analysis on the centers of geometry for whole residues. These values are plotted against their respective atom/residue number. They can also be converted to their beta-factor equivalents for direct comparison to crystallographic data. Users can choose to directly overlay up to three trajectories on one plot. Trajectories are aligned prior to analysis.

2.3.5 **Clustering**

Clustering is implemented using MDAnalysis (15) and allows users to choose from the AffinityPropagationNative or DBSCAN algorithms. Centroids are representative frames of a given conformational state and are saved as PDB files for the user's further consideration. Trajectories are aligned prior to analysis.

2.3.6 **Principal Component Analysis (PCA)**

PCA identifies arbitrary and orthogonal vectors that explain portions of the variation in a given dataset. The dot product of a frame onto these vectors projects the cartesian coordinates of every atom into principal component space, with as many dimensions as components used (26). Biotite-tools uses MDAnalysis (15) to perform this computation and always projects each frame of a trajectory onto the first two components (which explain the most variation). These projections are then displayed as a heatmap using gaussian interpolation implemented by matplotlib (13). The ranges of this heatmap can be adjusted by the user. Additionally, the kmeans2 clustering algorithm implemented by the scientific computing package scipy (22) or the MeanShift clustering algorithm implemented by the machine learning package scikit-learn (23) can be applied to these projections. Biotite-tools then highlights the frames closest to the identified centroids and saves them as PDB files for further analysis. The user can also specify specific frames to highlight on the heatmap. Multiple trajectories can be compared by projecting each onto the two components calculated from the first simulation. Trajectories are aligned prior to analysis.

2.3.7 **Hydrogen Bonding**

Hydrogen bonds are identified using MDAnalysis (15), which specifies potential acceptors and donors within 3 angstroms of one another and with a sufficiently large bonding angle. Biotite-tools allows users to specify which selection of atoms to check against which other selection of atoms. The generated data is saved as a table using NumPy. Multiple trajectories can be compared to identify which hydrogen bonds were relatively conserved across simulations and which were unique. Trajectories are not aligned to reduce computing time.

2.4 **Evaluating Biotite-tools Accuracy**

Because biotite-tools is a software that links pre-existing codebases to one another to enhance overall usability, it is challenging to quantify its "accuracy." While most of biotite-tools' generated figures and analyses were checked against alternative tools, such as Visual Molecular Dynamics (27), not all of them lend themselves to direct evaluation. We chose to demonstrate its effectiveness instead through a trial application to novel data. We used an unpublished simulation of the LARP1 protein's DM15 region to evaluate the efficiency and helpfulness of biotite-tools (28). Biotite-tools greatly accelerated the analysis of two simulations of alternative LARP1 DM15 crystal structures. The RMSF script in particular eased the identification of a number of vital residue motions that drive forward a mechanistic shift in the region's binding pocket. An example of how biotite-tools was applied to the LARP1 data can be seen in Figures 3-6.

3.0 Results and Discussion

3.1 **Dimorphite-DL's Empirical Approach**

Predicting the protonation state of a given compound is vital to VS and can mean the difference between identifying high and low binding affinity. Because most molecules contain ionizable sites (29), it is vital to account for the protonation state of each for a given pH range. Handling a single molecule containing N singly ionizable sites presents 2^N different possible structures. To enumerate every possible structure for a library of thousands of molecules would greatly delay the computer-aided drug design process. Dimorphite-DL addresses this challenge by assigning each site its own protonation state, ensuring only relevant structures remain to be screened.

Dimorphite-DL thus demonstrates the effectiveness of applying pre-existing experimental data to guide computational inquiry. Its simplicity ensures speedy execution, allowing it to scale to large libraries of compounds. Most other protonation prediction software operate by calculating quantum-mechanical interactions and predicting precise pK_a values. While this approach accounts for the structure of whole molecules in assigning protonation states, these programs are too slow for large datasets. Most are commercial and cannot be accessed for free nor actively improved upon by the open-source community (7).

3.2 **Dimorphite-DL Accuracy**

Dimorphite-DL's accuracy was independently evaluated for each moiety using 3-fold cross validation to prevent training bias. Potential outcomes were broken down into correct, excess, and incorrect predictions. As seen in Appendix A, the five most prominent moieties (boxed for convenience) within the training set depict high correct and excess percentages, demonstrating the effectiveness of our empirical algorithm. An excess prediction does not reduce the accuracy of the output; it only increases the number of generated structures and thus reduces efficiency. Users can offset this loss of efficiency for more precise predictions.

Because dimorphite-DL associates a pK_a range with each moiety, a user-provided parameter that scales this range directly affects the likelihood of two states being produced. Table 2 presents the impact of this parameter on the precision of our algorithm as shown by accuracy over all 38 structures (16). Notably, these percentages are over the entirety of the training set and are not a product of 3-fold cross validation. They serve to guide users in selecting an optimal precision. Adjusting this parameter fine-tunes dimorphite-DL to generate fewer or more protonation states. A high parameter increases the number of output structures, while a low parameter increases risk of producing incorrect states.

Table 2 Percent correct, excess, and incorrect predictions calculated as a weighted average across all moieties at increments of 0.5 from 0.0-3.0 precisions factor.

pK_a Precision Factor, *n* (Standard Deviations)	Correct (%)	Excess (%)	Incorrect (%)
0.0	70.9	23.9	5.2
0.5	69.1	26.5	4.4
1.0	58.8	40.2	0.9
1.5	51.2	48.8	0.0
2.0	50.7	49.3	0.0
2.5	23.9	76.1	0.0
3.0	22.1	77.9	0.0

3.3 **Biotite-tools**

Developing an empirical way to predict the protonation states of potential ligands is meaningless if they are docked into a protein structure that does not equally account for physiological relevance. MD simulations ensure this physiological relevance and it is just as important to process them in a simple but quick manner (8).

Biotite-tools aims to link the analytical functions of the MDAnalysis (15) library to publication quality visualizations through a customizable and easy-to-use command-line interface. The toolkit is provided as a collection of Python scripts that each provide their own specific analytical function and visualization. A user parameter system ensures that the scripts retain the bulk of the customizability of the functions that drive them.

3.3.1 **Aligning Trajectories**

MD simulation generally maintain physiological conditions, and as such proteins are often placed in a virtual box containing water molecules and *in* vivo ionic concentrations. As the MD engine calculates classical interatomic forces and applies them to the system, the protein naturally moves around this box, in addition to undergoing conformational fluctuation. Both conformational and translational motion occur. In order to only consider residue motion relative to the protein, we reassign protein coordinates at each frame by minimizing the distances between some set of atoms. Most frequently, we calculate this minimization by the alpha carbons of each residue. While biotite-tools provides this as a standalone script, it is vital to align a protein for most other analyses.

3.3.2 **Pruning and Trimming Trajectories**

MD simulations begin by assigning velocities to each atom to bring the static structure to a specified temperature. Due to the stochastic nature of this process, it is not uncommon for unstable interactions to result. We diminish these interactions by bringing a system to an equilibrium of distributed atomic energies. This prevents unwieldy forces from disturbing the protein. We use an iterative process through which a protein and its environment are gradually equilibrated. Upon viewing a completed simulation, it sometimes becomes apparent the system did not fully equilibrate. In this situation some portion of the simulation is still unrealistic and must be removed from the trajectory.

To speed up further analysis we also stride a trajectory by only considering a selection of evenly spaced frames. Taking every hundredth frame still provides ample biological insight while greatly reducing the runtime of performing statistical analysis.

3.3.3 **Root Mean Square Deviation (RMSD)**

RMSD reflects the overall motion that a group of atoms has undergone relative to a reference structure at each frame of a trajectory. Biotite-tools takes the first frame of a simulation as the reference in order to plot how much the protein has moved at each frame since beginning its trajectory (see Figure 3 for an example). The RMSD value only indicates how different a frame is from the reference structure; two frames with the same RMSD are not necessarily the same conformation, nor does the plot provide any atomic-resolution detail.

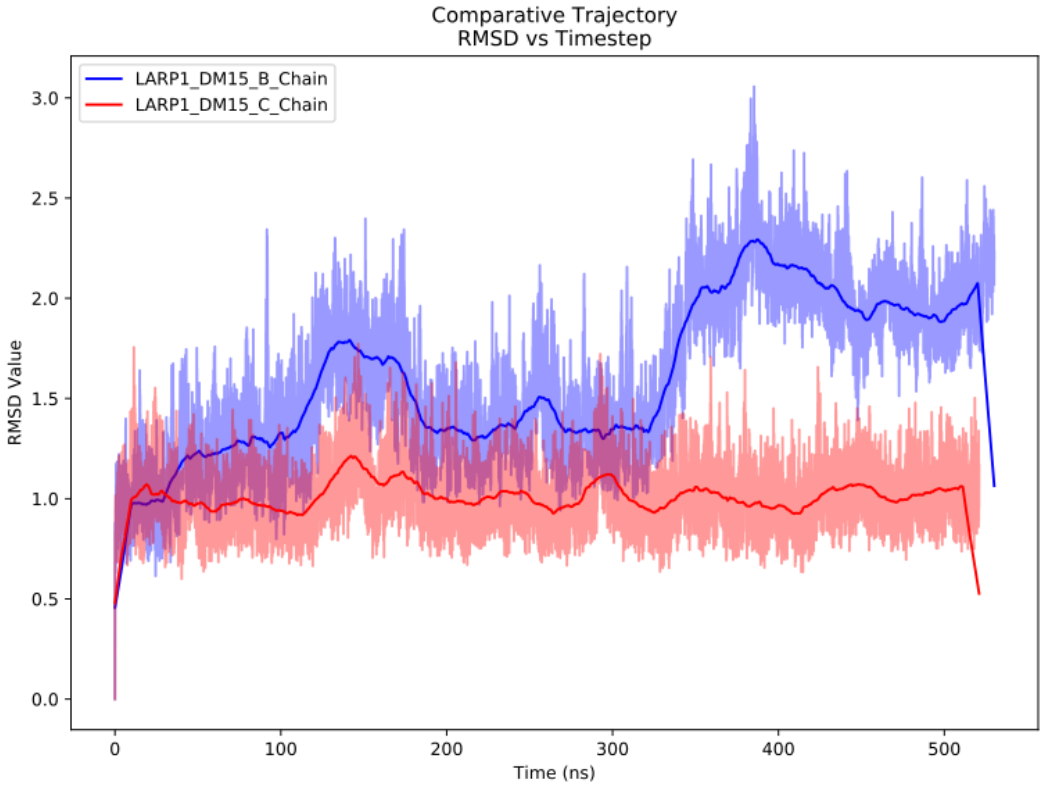


Figure 3 Root Mean Square Deviation comparative analysis of two LARP1 DM15 region MD simulations The two plots are overlaid and running averages are displayed as opaque lines. This plot demonstrates how biotite-tools revealed that the B chain (blue) simulation presented two distinct conformational states (28).

3.3.4 **Root Mean Square Fluctuation**

While RMSD reflects the motion of a whole group of atoms relative to time, RMSF reflects the motion of a specific atom or residue over the course of the entire simulation. RMSF provides a way to quickly visualize which atoms or residues experience more motion than others and enables a more informed viewing of a simulation. RMSF enables speedy identification of the atoms

responsible for an observed conformational shift. Plots (see figure 4) and PDB files (see figure 5) direct the user's analysis.

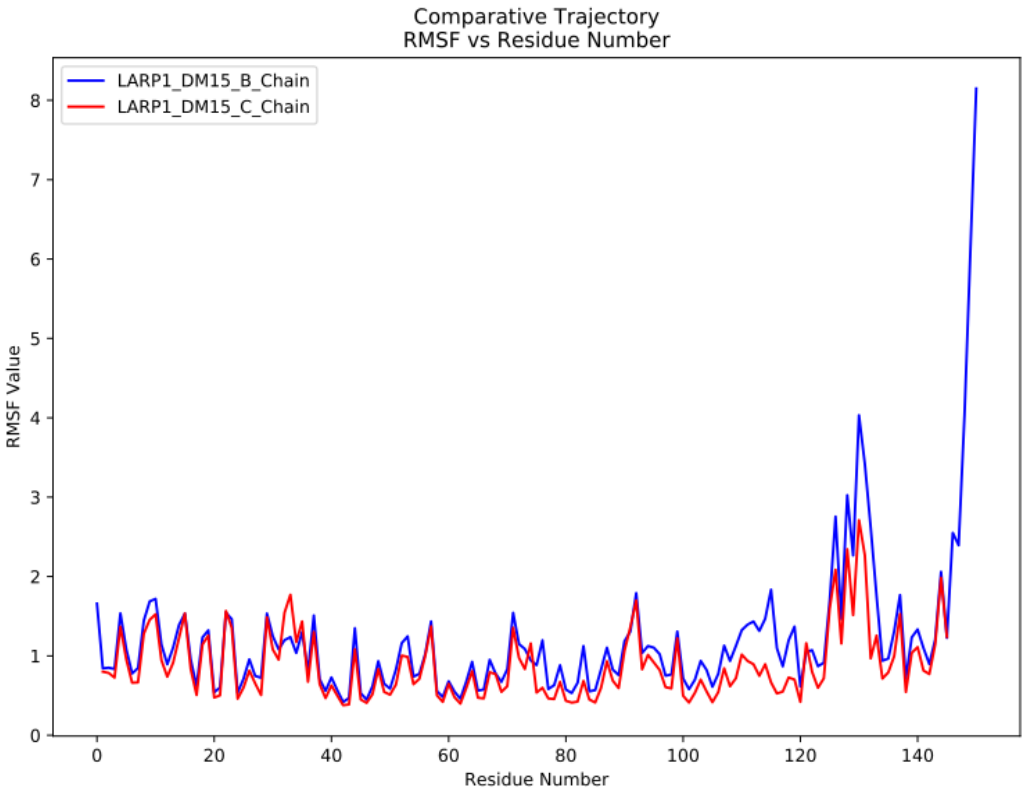


Figure 4 Root mean square fluctuation comparative analysis of two LARP1 DM15 region MD simulations The two plots are overlaid and visualize which residues experienced more motion throughout each simulation. This plot demonstrates how biotite-tools assisted in identifying which residues were responsible for the formation of a conformation only seen in the B chain simulation (28).

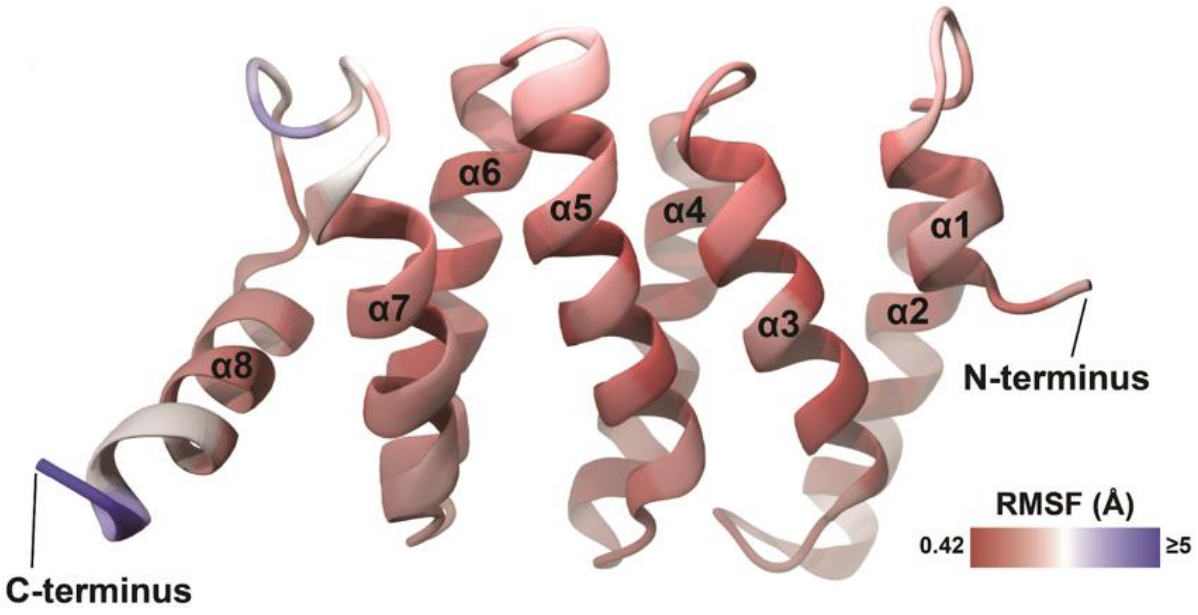


Figure 5 Root mean square fluctuation values projected directly onto the LARP1 DM15 region structure The PDB generated by biotite-tools was further visualized and rendered using BlendMol (33). The inter-helical loop reflects the highest non-terminal RMSF value. Analysis of this region has yielded novel insight into LARP1's mRNA binding mechanism (28). This was generated from the B chain simulation.

3.3.5 **Clustering**

The primary goal of MD is to identify the major energetic states that a protein occupies. While simulations provide visualizations of proteins transitioning between these conformations, it can be challenging to identify which frame of a simulation is representative of an energetic state. This is comparable to trying to identify which *specific* frame of a movie is most "representative" of a scene. Clustering algorithms provide a way to differentiate between these states and resolve them for further analysis and docking.

3.3.6 **Principal Component Analysis (PCA)**

A trajectory consists of the coordinates of each atom in cartesian space over time. The high dimensionality of these systems makes it a challenge to directly observe the path they take. PCA has established itself as a leading form of dimensionality-reduction that minimizes loss of variation. By transforming whole frames of a trajectory onto two principal components, biotite-tools can approximate a simulation in two dimensions as a heatmap (see Figure 6 for an example).

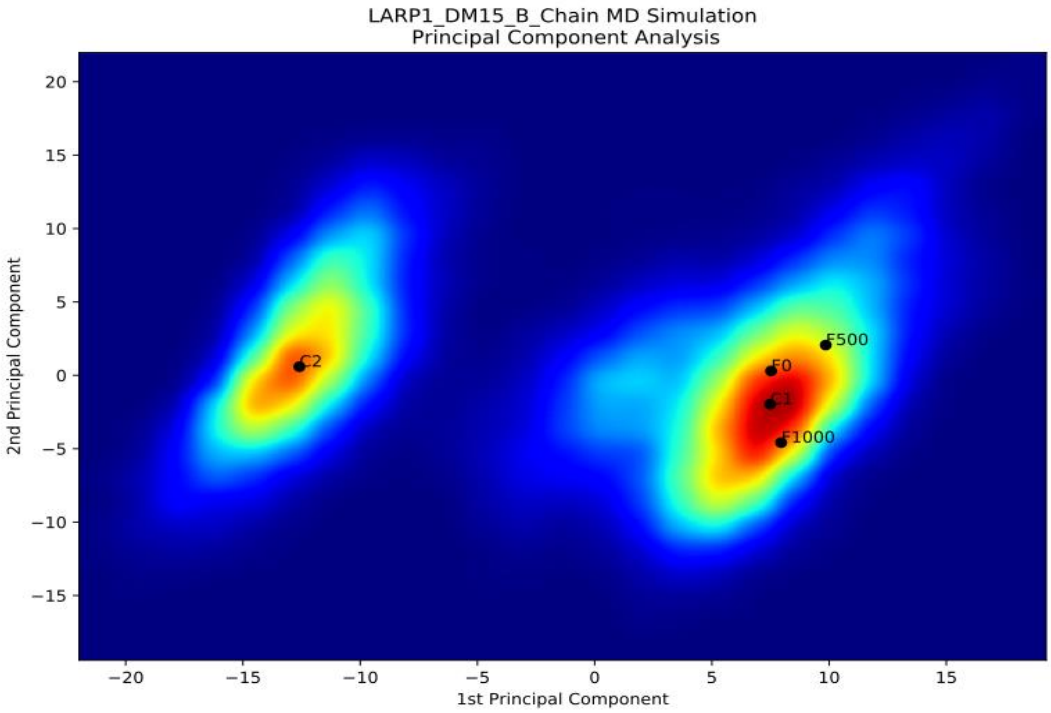


Figure 6 Principal Component Analysis of the LARP1 DM15 region B chain MD simulation
PCA is displayed as a heatmap using gaussian interpolation. Two distinct states are represented by the C1 and C2 centroids. Frames 0, 500, and 1000 are all localized in one conformational state, indicating that the protein did not shift into an alternative energy well until the latter half of the trajectory, matching observations in Figure 3.

Such a heatmap provides a categorization of the overall energetics of a simulated protein. Because the heatmap is colored by the density of frames with unique coordinates on the two components, hotspots indicate *distinct* energetic states and exchange temporal resolution for the

conformational specificity lacking in RMSD analysis. The ability to highlight representative (determined by clustering) and pre-specified frames helps compensate for the intrinsic loss of temporal information in PCA.

3.3.7 **Hydrogen Bonding**

Hydrogen bonds play a major role in mediating macromolecular tertiary structure (30). Identifying where and how these hydrogen bonds form is important to understanding the stabilization and driving forces behind conformational shifts (see LARP1 analysis below).

3.3.8 **Comparative Analysis**

MD simulations calculate forces deterministically, however they are highly parameterized and multiple simulations of the same protein can take very different routes depending on the chosen environment and initial assignment of atomic velocities (8). In simulating proteins of interest, we often perform one lengthy simulation, as well as two smaller ones generated from the point of equilibration. The hope is to sample as many energetic paths as possible, while allowing for any motion that may only occur on microsecond timescales. This approach necessitates directly comparing multiple simulations of the same structure. Simulating the same protein in two different environments, for example in the presence of higher ionic concentrations or small organic molecules, also requires comparative analysis.

Every biotite-tools script takes in batch as many trajectories as provided and can optionally perform a comparative analysis. For some, like RMSD and RMSF, this may simply mean overlaying the two plots to visualize the differences between them. For the more complex process

of PCA, it is vital that each trajectory be projected onto the same components. Calculating components from each trajectory independently would render any comparison meaningless. It is difficult to predict all the different ways a user might want to compare hydrogen bonding. Biotite-tools takes a general approach, informing the user which bonds were unique to specific simulations and which were shared across trajectories.

3.3.9 **LARP1 demonstrates the need for a server implementation**

An analysis of the LARP1 protein's DM15 domain using biotite-tools found a novel mechanism behind its binding TOP-mRNA transcripts (28). By comparing RMSD and RMSF values, an inter-helical loop was identified that transiently enters a helix structure to enlarge the binding pocket for transcripts. Several hydrogen-bonding interactions between proximal residues mediate this process. Clustering in conjunction with binding pocket analysis using the FTMap server (31) identified novel conformations. LARP1's regulation of translational machinery mRNA transcripts is implicated in various cancers (32), so selective targeting of this protein has important implications for future treatment. This analysis demonstrates how biotite-tools and other accessible bioinformatic tools can drive forward the drug discovery process.

This proof of concept with LARP1 provides an example of how biotite-tools as a standalone entity can accelerate the process of those already familiar with programming. However, much of the original code in biotite-tools exists to bring together the three worlds of MDAnalysis' analytical functions, matplotlib's plotting capability, and user interaction. The challenge is how to present these analyses in a manner intuitive to users without loss of customizability, while also targeting the scripts at implementation on a server-hosted GUI. A well-defined user parameter system provides the customizability that, in conjunction with extensive documentation,

simultaneously guides the user in analysis. Several implementation details allow biotite-tools to be adapted to a browser-based system. The primary future direction for these scripts is incorporation into a work-in-progress server that will also guide the running of MD simulations and further analyses such as druggability and docking, for which excellent codebases already exist (3).

4.0 Conclusion

Here we presented two open-source programs, dimorphite-DL and biotite-tools, targeted at accelerating structure-based drug design. Dimorphite-DL uses empirically determined pK_a values to predict the protonation states of novel molecules. This ensures more accurate and efficient VS to identify potential pharmaceuticals. Biotite-tools provides a suite of easy-to-use scripts for analyzing MD simulations and generating publication-ready figures. Dimorphite-DL and biotite-tools are built for both independent use and future incorporation into larger projects. Dimorphite-DL demonstrates the advantages of a close collaboration between the worlds of experimental and theoretical biology, while biotite-tools directly draws on pre-existing code to enhance usability in a way that prioritizes future development.

The open-source movement has already established itself strongly in academia. Because the consequences of novel biomedical findings are life and death matters for those who need them, any obstacle to their discovery must be avoided. Developing code readily accessible and openly contributing to a communal knowledge-base is the most effective way to accelerate research. Biomedical tools are not a product of commercial interest nor career advancing motivations, but of the recognition of a physiological or methodological problem. Because these challenges necessarily arise in the course of research, those who identify these problems have a responsibility to ensure their resolution, be it through their own development or through the open-source community and collaborative outreach. Placing a restriction on a biomedical tool, be it through licensing, vague documentation, or lack of accessibility, impedes solutions to medical challenges.

Appendix A Dimorphite-DL Prediction Accuracy

Dimorpite-DL protonation prediction accuracy for 37 of 38 supported moieties at precision factor 0, 1.0, 1.5, and 2.0 (16). The five moieties most present within our training data are boxed.

		stdev: 0.0	stdev: 1.0	stdev: 1.5	stdev: 2.0
Alcohol	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Amide	Correct (%)	83.3 ± 13.6	22.2 ± 31.4	5.6 ± 7.9	5.6 ± 7.9
	Excess (%)	0.0 ± 0.0	66.7 ± 47.1	94.4 ± 7.9	94.4 ± 7.9
	Incorrect (%)	16.7 ± 13.6	11.1 ± 15.7	0.0 ± 0.0	0.0 ± 0.0
Amide_electronegative	Correct (%)	84.3 ± 4.6	59.3 ± 33.3	7.9 ± 5.6	7.9 ± 5.6
	Excess (%)	0.0 ± 0.0	29.2 ± 41.2	92.1 ± 5.6	92.1 ± 5.6
	Incorrect (%)	15.7 ± 4.6	11.6 ± 9.1	0.0 ± 0.0	0.0 ± 0.0
AmidineGuanidine1	Correct (%)	93.3 ± 9.4	93.3 ± 9.4	93.3 ± 9.4	60.0 ± 43.2
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	33.3 ± 47.1
	Incorrect (%)	6.7 ± 9.4	6.7 ± 9.4	6.7 ± 9.4	6.7 ± 9.4
AmidineGuanidine2	Correct (%)	75.5 ± 3.9	21.5 ± 4.1	21.5 ± 4.1	21.5 ± 4.1
	Excess (%)	0.0 ± 0.0	78.5 ± 4.1	78.5 ± 4.1	78.5 ± 4.1
	Incorrect (%)	24.5 ± 3.9	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Amines_primary_secondary_tertiary	Correct (%)	26.9 ± 3.0	26.9 ± 3.0	26.9 ± 3.0	26.9 ± 3.0
	Excess (%)	73.1 ± 3.0	73.1 ± 3.0	73.1 ± 3.0	73.1 ± 3.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Anilines_primary	Correct (%)	94.8 ± 3.7	61.4 ± 43.4	30.7 ± 43.4	0.0 ± 0.0
	Excess (%)	0.0 ± 0.0	33.3 ± 47.1	66.7 ± 47.1	100.0 ± 0.0
	Incorrect (%)	5.2 ± 3.7	5.2 ± 3.7	2.7 ± 3.8	0.0 ± 0.0
Anilines_secondary	Correct (%)	83.7 ± 2.7	14.2 ± 2.4	14.2 ± 2.4	14.2 ± 2.4
	Excess (%)	0.0 ± 0.0	85.8 ± 2.4	85.8 ± 2.4	85.8 ± 2.4
	Incorrect (%)	16.3 ± 2.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Anilines_tertiary	Correct (%)	84.2 ± 8.6	84.2 ± 8.6	14.0 ± 6.6	14.0 ± 6.6
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	86.0 ± 6.6	86.0 ± 6.6
	Incorrect (%)	15.8 ± 8.6	15.8 ± 8.6	0.0 ± 0.0	0.0 ± 0.0
Aromatic_nitrogen_protonated	Correct (%)	75.0 ± 10.2	8.3 ± 5.9	8.3 ± 5.9	8.3 ± 5.9
	Excess (%)	0.0 ± 0.0	91.7 ± 5.9	91.7 ± 5.9	91.7 ± 5.9
	Incorrect (%)	25.0 ± 10.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Aromatic_nitrogen_unprotonated	Correct (%)	88.0 ± 2.7	35.1 ± 36.2	9.2 ± 3.5	9.2 ± 3.5
	Excess (%)	0.0 ± 0.0	60.2 ± 42.7	90.8 ± 3.5	90.8 ± 3.5
	Incorrect (%)	12.0 ± 2.7	4.6 ± 6.5	0.0 ± 0.0	0.0 ± 0.0
Carboxyl	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Imide	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	66.7 ± 47.1
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	33.3 ± 47.1
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Imide2	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	66.7 ± 47.1	33.3 ± 47.1
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	33.3 ± 47.1	66.7 ± 47.1
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
N-hydroxyamide	Correct (%)	61.9 ± 33.7	14.3 ± 0.0	38.1 ± 33.7	38.1 ± 33.7
	Excess (%)	0.0 ± 0.0	57.1 ± 40.4	61.9 ± 33.7	61.9 ± 33.7
	Incorrect (%)	38.1 ± 33.7	28.6 ± 40.4	0.0 ± 0.0	0.0 ± 0.0
O=C-C=C-OH	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Peroxide1	Correct (%)	72.2 ± 20.8	27.8 ± 20.8	27.8 ± 20.8	27.8 ± 20.8
	Excess (%)	0.0 ± 0.0	72.2 ± 20.8	72.2 ± 20.8	72.2 ± 20.8

	Incorrect (%)	27.8 ± 20.8	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Peroxide2	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phenol	Correct (%)	33.7 ± 3.8	33.7 ± 3.8	33.7 ± 3.8	33.7 ± 3.8
	Excess (%)	66.3 ± 3.8	66.3 ± 3.8	66.3 ± 3.8	66.3 ± 3.8
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phenyl_carboxyl	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phenyl_thiol	Correct (%)	77.8 ± 15.7	16.7 ± 13.6	16.7 ± 13.6	16.7 ± 13.6
	Excess (%)	0.0 ± 0.0	83.3 ± 13.6	83.3 ± 13.6	83.3 ± 13.6
	Incorrect (%)	22.2 ± 15.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phosphate	Correct (%)	70.4 ± 5.2	63.0 ± 13.9	63.0 ± 13.9	63.0 ± 13.9
	Excess (%)	18.5 ± 13.9	37.0 ± 13.9	37.0 ± 13.9	37.0 ± 13.9
	Incorrect (%)	11.1 ± 15.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phosphate_diester	Correct (%)	95.2 ± 6.7	95.2 ± 6.7	28.6 ± 40.4	28.6 ± 40.4
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	66.7 ± 47.1	66.7 ± 47.1
	Incorrect (%)	4.8 ± 6.7	4.8 ± 6.7	4.8 ± 6.7	4.8 ± 6.7
Phosphinic_acid	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phosphonate	Correct (%)	76.5 ± 14.4	76.5 ± 14.4	76.5 ± 14.4	76.5 ± 14.4
	Excess (%)	23.5 ± 14.4	23.5 ± 14.4	23.5 ± 14.4	23.5 ± 14.4
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Phosphonate_ester	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Primary_hydroxyl_amine	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ringed_imide1	Correct (%)	11.1 ± 15.7	55.6 ± 41.6	55.6 ± 41.6	55.6 ± 41.6
	Excess (%)	33.3 ± 47.1	44.4 ± 41.6	44.4 ± 41.6	44.4 ± 41.6
	Incorrect (%)	55.6 ± 41.6	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ringed_imide2	Correct (%)	58.3 ± 21.2	12.5 ± 17.7	12.5 ± 17.7	12.5 ± 17.7
	Excess (%)	20.8 ± 29.5	87.5 ± 17.7	87.5 ± 17.7	87.5 ± 17.7
	Incorrect (%)	20.8 ± 21.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Sulfate	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Sulfinic_acid	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Sulfonamide	Correct (%)	37.1 ± 11.6	37.1 ± 11.6	37.1 ± 11.6	37.1 ± 11.6
	Excess (%)	62.9 ± 11.6	62.9 ± 11.6	62.9 ± 11.6	62.9 ± 11.6
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Sulfonate	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Thioic_acid	Correct (%)	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Excess (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Incorrect (%)	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Thiol	Correct (%)	61.9 ± 6.0	38.1 ± 6.0	38.1 ± 6.0	38.1 ± 6.0
	Excess (%)	0.0 ± 0.0	61.9 ± 6.0	61.9 ± 6.0	61.9 ± 6.0
	Incorrect (%)	38.1 ± 6.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Vinyl_alcohol	Correct (%)	81.0 ± 6.7	9.5 ± 6.7	9.5 ± 6.7	9.5 ± 6.7
	Excess (%)	0.0 ± 0.0	90.5 ± 6.7	90.5 ± 6.7	90.5 ± 6.7
	Incorrect (%)	19.0 ± 6.7	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Average	Correct (%)	81.3	64.1	56.5	52.9
	Excess (%)	8.3	33.5	43.1	46.8
	Incorrect (%)	10.4	2.5	0.4	0.3

Bibliography

1. Bohacek, R.S., C. McMartin, and W.C. Guida, The art and practice of structure-based drug design: a molecular modeling perspective. Med Res Rev, 1996. **16**(1): p. 3-50.

2. Kitchen, D.B., et al., Docking and scoring in virtual screening for drug discovery: methods and applications. Nat Rev Drug Discov, 2004. **3**(11): p. 935-49.

3. Trott, O. and A.J. Olson, AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. J Comput Chem, 2010. **31**(2): p. 455-61.

4. Knox, A.J., et al., Considerations in compound database preparation--"hidden" impact on virtual screening results. J Chem Inf Model, 2005. **45**(6): p. 1908-19.

5. Rapp, C.S., et al., Automated site preparation in physics-based rescoring of receptor ligand complexes. Proteins, 2009. **77**(1): p. 52-61.

6. Petukh, M., S. Stefl, and E. Alexov, The role of protonation states in ligand-receptor recognition and binding. Curr Pharm Des, 2013. **19**(23): p. 4182-90.

7. Liao, C. and M.C. Nicklaus, Comparison of nine programs predicting pK(a) values of pharmaceutical substances. J Chem Inf Model, 2009. **49**(12): p. 2801-12.

8. Durrant, J.D. and J.A. McCammon, Molecular dynamics simulations and drug discovery. BMC Biol, 2011. **9**: p. 71.

9. Wang, J., et al., Development and testing of a general amber force field. J Comput Chem, 2004. **25**(9): p. 1157-74.

10. McCammon, J.A., B.R. Gelin, and M. Karplus, Dynamics of folded proteins. Nature, 1977. **267**(5612): p. 585-90.

11. Chodera, J.D., et al., Alchemical free energy methods for drug discovery: progress and challenges. Curr Opin Struct Biol, 2011. **21**(2): p. 150-60.

12. McGibbon, R.T., et al., MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. Biophys J, 2015. **109**(8): p. 1528-32.

13. John D. Hunter. **Matplotlib: A 2D Graphics Environment**, Computing in Science & Engineering, **9**, 90-95 (2007),DOI:10.1109/MCSE.2007.55

14. Wes McKinney. **Data Structures for Statistical Computing in Python**, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

15. Michaud-Agrawal, N., et al., MDAnalysis: a toolkit for the analysis of molecular dynamics simulations. J Comput Chem, 2011. **32**(10): p. 2319-27.

16. Ropp, P.J., et al., Dimorphite-DL: an open-source program for enumerating the ionization states of drug-like small molecules. J Cheminform, 2019. **11**(1): p. 14.

17. Lee, A.C., J.Y. Yu, and G.M. Crippen, pKa prediction of monoprotic small molecules the SMARTS way. J Chem Inf Model, 2008. **48**(10): p. 2042-53.

18. Internet Bond-Energy Databank, T.a.N. Universities. http://ibond.nankai.edu.cn/. Accessed November 13, 2018.

19. Weininger, D., SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comp Sci, 1988. **28**: p. 31-36.

20. Landrum, G., RDKit: open-source cheminformatics. http://www.rdkit.org/. Accessed November 13, 2018.

21. Travis E, Oliphant. **A Guide to NumPy**, USA: Trelgol Publishing, (2006).

22. Jones E, Oliphant E, Peterson P, *et al.* **SciPy: Open Source Scientific Tools for Python**, 2001, http://www.scipy.org/ [Online; accessed 2019-03-04].

23. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. **Scikit-learn: Machine Learning in Python**, Journal of Machine Learning Research, **12**, 2825-2830 (2011)

24. Theobald, D.L., Rapid calculation of RMSD using a quaternion-based characteristic polynomial. Acta Crystallographica A, 2005. **61**(4): p. 478-480.

25. Welford, B.P., Note on a Method for Calculating Corrected Sums of Squares and Products. Technometrics, 1962. **4**(3): p. 419-420.

26. Joliffe, I.T. and B.J. Morgan, Principal component analysis and exploratory factor analysis. Stat Methods Med Res, 1992. **1**(1): p. 69-95.

27. Humphrey, W., A. Dalke, and K. Schulten, VMD: visual molecular dynamics. J Mol Graph, 1996. **14**(1): p. 33-8, 27-8.

28. Cassidy, K.C., et al., Capturing the Mechanism Underlying TOP-mRNA binding to the LARP1-specific DM15 Region. Unpublished data.

29. Greenwood, J.R., et al., Towards the comprehensive, rapid, and accurate prediction of the favorable tautomeric states of drug-like molecules in aqueous solution. J Comput Aided Mol Des, 2010. **24**(6-7): p. 591-604.

30. Dill, K.A., Dominant forces in protein folding. Biochemistry, 1990. **29**(31): p. 7133-55.

31. Kozakov, D., et al., The FTMap family of web servers for determining and characterizing ligand-binding hot spots of proteins. Nat Protoc, 2015. **10**(5): p. 733-55.

32. Mura, M., et al., LARP1 post-transcriptionally regulates mTOR and contributes to cancer progression. Oncogene, 2015. **34**(39): p. 5025-36.

33. Durrant, J.D., BlendMol: Advanced Macromolecular Visualization in Blender. Bioinformatics, 2018.