SilcsBio User Guide

Release 2026.1

May 2026

CONTENTS

SilcsBio LLC
1300 Bayard Street
Baltimore, MD 21230
info@silcsbio.com
https://silcsbio.com

CHAPTER

ONE

ABOUT THIS DOCUMENT

SilcsBio offers a suite of computer-aided drug design software including:

- the patented SILCS functional group mapping approach with capabilities for
  (a) small-molecule database screening, fragment-based drug design, and lead optimization,
  (b) covalent warhead screening and covalent inhibitor optimization, and
  (c) excipient screening for biomacromolecular therapeutics;
- the SSFEP method that delivers free energy calculations 1,000 times faster than standard FEP; and
- the CGenFF force field parameter assignment engine.

This documentation covers how to install and use the software.

CHAPTER

TWO

RELEASE NOTES

2.1 Version 2026.1

This version adds the following:

- New SilcsBio GUI updates
  - Support for visualization of multiple poses of Docked Ligands
  - Improved cluster-side directory naming convention to include project name for easier identification
  - Improved restart mechanics for SILCS simulations, allowing user to pull progress report and log file information direct to GUI
- New CLI additions
  - Kinetics Suite - New Release
    - SILCS-Pathway: a new module to identify ligand binding/unbinding pathways using SILCS-FragMaps and A* search algorithm.
    - SILCS-Kinetics: a new module to estimate kinetic parameters of ligand binding/unbinding using the pathways identified by SILCS-Pathway and the free energy profiles along the pathways.
  - CGenFF
    - Support for batch processing of ligand files for CGenFF parameterization available in individual and combined modes
    - CGenFF updated to version 5.0
    - GROMACS conversion now creates a single .top file merging the ff parameters and topology for the ligand
  - SILCS simulations
    - Support for extracting frames from fragmap simulations where SILCS probes are within a user-defined distance from a specified residue.

  - SILCS-Biologics
    * New scripts to aid in post-processing of SILCS-Biologics results, including generation of plots for binding predictions, and a script to generate a series of SILCS-Biologics results for different user-inputed formulations with experimental data
    * Update to SILCS-Biologics log file naming convention includes step identification to prevent overwriting of log files and to facilitate identification of log files for each step of the workflow
  - SILCS-Covalent
    * Support for sdf ligand input
    * Retain cutoffs used in data collection for user record
  - SILCS-Hotspots
    * Added amino acid databases to SILCS-Hotspots. Inlcuded are dual-capped single amino acids, dual-capped amino acid pairs, and amino acids side chains
- CGenFF Web App
  - Updated to version 5.0
  - Allows for version selection between v 5.0 and v 4.6

as well as bug fixes and stability improvements.

2.2 Version 2025.1

This version adds the following:

- New SilcsBio GUI updates
  - Completed projects can be exported/imported accross users
  - Remaining time estimates are displayed for each SILCS simulation
  - Pharmacophore features are labeled to facilitate viewing
  - SILCS-MC docking accepts a directory of ligand files in MOL2 or SDF format
  - SILCS-MC docking allows for GPU or CPU usage
  - LGFE values are displayed in the SILCS-MC docking results chart tab
- New CLI additions
  - SILCS simulations
    * Additional restraint modes for SILCS simulations
    * Support for multiple ligands/cofactors - both standard and covalent

        * Additional bilayers compositions for membrane protein setup
        * Protein side chain probability maps (ProtMaps) from SILCS simulations
    - SILCS-Hotspots
        * ML-based extension for more accurate identification and ranking of hotspots
    - SILCS-MC Docking
        * Support for ligand alignment to reference ligand prior to MCSA docking
        * RMSD-clustering of SILCS-MC docking results
        * Option to write RMSD w.r.t. input pose to the lgfe.csv file
    - SILCS-MC Ligand Optimization
        * Generate a database with ligand analysis calculation for modifications to show LGFE contributions from common and modified substructures in comparison with parent ligand
    - SILCS-WATER
        * Dock ligands in the presence of water molecules
        * Dock multiple ligands in a single run taking interaction between ligands into account
    - SILCS-Pharm
        * Standalone script to prepare PHARMER-compatible databases.
    - SILCS-Biologics
        * Estimate protein effective charge accounting for ions, buffers, and excipients
        * Generate a series of formulation with different combinations of excipients and excipient concentrations
- SILCS-Covalent: a new module to support covalent drug design.

as well as bug fixes and stability improvements.

2.3 Version 2024.2

This version adds the following updates to the SilcsBio GUI only:

- New SilcsBio GUI updates
    - Improvements in FragMap visualization
    - Recent visualization sessions are stored in for easy access

    - Separate FragMap and Ligand window consolidated into sidebar menu
    - FragMap checkbox colors correspond to FragMap isosurface colors
    - Reset rotation, translation, and scaling with a click of a button
  - Native file browser of local OS is used for local files
  - SILCS-MC pose collection: poses will be collected in both PDB and SD file formats

2.4 Version 2024.1

This version adds the following:

- SILCS-MC using GPU (Nvidia CUDA), performance increased by 700%
- SILCS-MC: added slab mode for docking along user-defined path
- New SilcsBio GUI additions
  - SILCS-RNA in GUI
  - SILCS-Membrane Protein in GUI
  - SILCS-MC Pose Evaluation in GUI
  - CGenFF Suite in GUI with basic functionality
- New CLI additions
  - SILCS-MC Pose Evaluation in CLI
  - Bayesian Machine Learning module (SILCS-BML) in CLI
  - SILCS trajectory scanning module for finding the protein conformations that best complement docked ligand poses in CLI
  - Pharmacophore screening (Pharmer) in CLI
  - CGenFF-covalent module in CLI
  - MD module in CLI
    * Standard MD: setup, run and analyze
    * MD with covalent ligand
    * MD with single probe
    * MD for membrane protein

as well as bug fixes and stability improvements.

2.5 Version 2023.1

This version adds the following:

- GCMC sampling utilizing GPU resources. Prior versions utilized GPU resources for the MD portion of the GCMC/MD sampling in SILCS simulations whereas GCMC sampling was done using CPUs only. The new GPU GCMC code can make runs 50% faster overall while also enabling more efficient probe sampling due to algorithmic improvements for mu_ex oscillation.
- SILCS protocol specifically tailored for RNA targets
- SILCS-Biologics workflow setup, run control, and data analysis from the SilcsBio Graphical User Interface (GUI)
- SILCS support for membrane proteins other than GPCRs, such as transporters and ion channels
- SILCS-Hotspots output validation capability by docking of a diverse subset of FDA-approved drugs
- Mechanism for checking internet connection, server connection, and SILCS server version compatibility from within the SilcsBio GUI
- Integrated SILCS-Biologics documentation access from within the SilcsBio GUI
- Updated graphical user interface design
- Compatibility with GROMACS version 2022.x
- Improved jobs control within the SilcsBio GUI
- Scripting to subtract one set of FragMaps from another, including automatic handling of differences in FragMap grid dimensions. This can be useful for qualitative analysis of functional group binding preferences between two homologous proteins.

as well as bug fixes and stability improvements.

2.6 Version 2022.2

This version adds the following:

- SilcsBio Graphical User Interface (GUI) support for Halogen SILCS simulation
- SilcsBio GUI support for inclusion of Halogen SILCS FragMaps in ligand posing and scoring for SILCS-MC: Docking and Pose Refinement and SILCS-MC: Ligand Optimization
- Support for custom temperature choice for SILCS simulations

- Support for using existing setup/trajectory files for starting SILCS simulations via the SilcsBio GUI
- Expanded SilcsBio GUI visualization capabilities including: atomic GFE values and their sums for ligands scored with SILCS-MC; comprehensive atom selection options for visualization; and residue labels on proteins
- Usability improvement to the SilcsBio GUI including: showing project creation dates; ability to sort by date or name during file/directory selection; and progress bar for file transfers
- CGenFF program is version 2.5.2 and CGenFF parameters are version 4.6

as well as bug fixes and stability improvements.

2.7 Version 2022.1

This version adds the following:

- SilcsBio Graphical User Interface (GUI) support for SILCS-Hotspots
- Distance measurement between SILCS-Pharm pharmacophore features in the SilcsBio GUI
- Charting results from SILCS-MC jobs (ligand optimization, docking, pose refinement, hotspots) prior to full job completion, and recalculation of results at any time through the SilcsBio GUI
- Selection of number of processors and number of independent runs for SILCS-MC through the SilcsBio GUI
- SILCS-Hotspots report generation script update to show additional measurements
- SILCS-MC docking and pose refinement expansion to include Halogen FragMaps
- SILCS simulation with iron ions can be with +2 (default) or +3 charge
- Dimethylether replaces acetaldehyde in Standard SILCS simulations
- CGenFF program update to version 2.5.1 and CGenFF parameters update to version 4.6

  The CGenFF program version 2.5.1 can optionally retain partial charges from the input Mol2 file instead of assigning them. CGenFF parameters version 4.6 is updated with training that adds numerous new compounds:

  Model compounds for nonstandard amino acids (206 new residues encompassing 87 new bond parameters, 437 new angle parameters, 1698 new dihedral terms, and 34 new improper dihedral parameters) [1]

  Flavin and flavin-related model compounds

as well as bug fixes and stability improvements.

2.8 Version 2021.1

This version adds the following:

- The SILCS-Biologics method for excipient screening for biomacromolecular therapeutics
- CGenFF program update to version 2.5 and CGenFF parameters update to version 4.5

  CGenFF coverage has been extended to the following compounds that have been explicitly parameterized and validated, allowing CGenFF to generate parameters for more diverse compounds.

  ASBB: (1-[(2-aminoethyl)sulfanyl]butan-1-one): charges and bonded parameters

  2-phenylthiazole and 5-methyl-3-phenyl-1,2,4-oxadiazole: bonded parameters

  Fentanyl: bonded parameters [2]

  (E)-1,2-di-p-tolyldiazene: bonded parameters [3]

as well as bug fixes and stability improvements.

2.9 Version 2020.2

This version adds the following:

- SilcsBio Graphical User Interface (GUI) improvements for file and directory selection allowing for input files to be chosen from remote servers as well as the local computer
- SilcsBio Graphical User Interface (GUI) support for visualization of Halogen SILCS FragMaps
- A new plug-in for visualizing SILCS FragMaps in MOE
- CGenFF program and parameters update to version 2.4.0

  CGenFF coverage has been extended to amide bases and molecules containing boron. The functional groups in these molecules were not previously accessible in CGenFF. The following compounds have been explicitly parameterized and validated, allowing CGenFF to generate parameters for more diverse compounds.

  N-methyl acetamide (deprotonated amide); N-ethyl acetamide (deprotonated amide); N-methyl benzamide (deprotonated amide); N-phenyl acetamide (deprotonated amide)

  Methyl boronic acid (neutral, -1, -2); Ethyl boronic acid (neutral, -1, -2); Phenyl boronic acid (neutral, -1, -2)

  Additionally, 112 naturally-occurring modified nucleotides, especially modified bases with heterocycles not previously covered by CGenFF, have been parametrized. Quantum mechanical calculations on model compounds, including geometries, dipole moments, and interac-

tions with water, provided target data. Selected parameters were validated with extensive molecular dynamics simulations. Details of the parameter optimization and the complete list of nucleotides can be found in [4].

as well as bug fixes and stability improvements.

Validation of the SILCS-HotSpots approach has been published [5].

2.10 Version 2020.1

This version adds the following:

- The SILCS-Hotspots method for identifying all potential ligand binding sites on a protein
- SilcsBio Graphical User Interface (GUI) support for SILCS-Pharm
- SilcsBio Graphical User Interface (GUI) support for SILCS-MC Docking and Pose Generation
- Performance improvements to GCMC-MD
- CGenFF program and parameters update to version 2.3.0

  CGenFF version 2.3.0 adds explicit parametrization for the following molecules. In prior versions, functional groups in these molecules were accessible through analogy to related functional groups. Now, explicit parametrization and validation yields more accurate treatment and decreases the associated CGenFF penalty scores.

  1H-tetrazole; 5-methyl-1H-tetrazole; 5-ethyl-1H-tetrazole

  2-oxetanone; 3-oxetanone

  ammonium; dimethylammonium; trimethylammonium (Note: Protonated amine parameters were previously based on methylammonium. While the present explicit parametrization of secondary and tertiary amines leads to smaller penalty scores and improvements in performance, electrostatic interactions will continue to be dominated by the +1 monopole.)

  1-butyne; 1-pentyne; 1-hexyne; 1-heptyne; 1-octyne; but-1-ene-3-yne (Note: Alkyne parmeters were previously based on ethene and propene. Extension to longer alkynes and the ene/yne combination validates the parameters and leads to improved treatment of the intramolecular interactions.)

  CGenFF version 2.3.0 also includes improved halogen–protein interactions. Quantum mechanical calculations on chloro- and bromobenzene with model compounds representative of protein functional groups were used as target data to optimize atom-pair specific Lennard-Jones parameters for selected atoms in the model compounds. Application of the parameters in molecular dynamics simulations of eight ligand-protein systems demonstrated systematic improvement in interaction geometries.

as well as bug fixes and stability improvements.

2.11 Version 2019.2

This version adds the following:

- The SILCS-Pharm method for generating 3D receptor-based pharamcophore models from FragMaps in an automated manner

as well as bug fixes and stability improvements.

2.12 Version 2019.1

This version adds the following:

- The SILCS-Pharm method for generating 3D receptor-based pharamcophore models from FragMaps in an automated manner (early access)

as well as bug fixes and stability improvements.

2.13 Version 2018.2

This version adds the following:

- A new Graphical User Interface (GUI) capable of preparing and launching SILCS, SILCS-MC, and SSFEP jobs on remote computing clusters and analyzing and visualizing the job outputs
- A new CGenFF Parameter Optimizer with functionality for dihedral parameter fitting

as well as bug fixes and stability improvements.

2.14 Version 2018.1

This version adds the following:

- A new Graphical User Interface (GUI) capable of preparing and launching SILCS, SILCS-MC, and SSFEP jobs on remote computing clusters and analyzing and visualizing the job outputs (early access)
- Improved GPCR and other transmembrane protein support for SILCS
- Improved GPCR and other transmembrane protein support for SSFEP
- CGenFF program and parameters update to version 2.2.0

CGenFF version 2.2.0 extends support to a large variety of drug-like molecules to be used routinely in computer-aided drug design projects. Specifically, 1) it improves treatment of halogen bonds by introducing lone-pairs onto the halogen atoms of aromatic systems. 2) Support is extended to four-membered oxetane and glycoluril. 3) Improved predictions with partial charge distributions around ammonium ions and primary amines.

- A new CGenFF Parameter Optimizer with functionality for dihedral parameter fitting (early access)

as well as bug fixes and stability improvements.

2.15 Version 2017.2

This version adds the following:

- GPCR support for SILCS
- GPCR support for SSFEP
- CGenFF program and parameter update to version 2.1.0

  CGenFF version 2.1.0 extends support to S-P bond found in the GTP-gamma like molecules. Bonded parameters and charge-distribution along the S-P bond were modeled using the CHARMM nucleic acid force-field patches for mono- and di-thio substitutions (top-par_all36_na_reactive_rna.str). Three new atom-types have been added to the CGenFF force-field : SG2P1, SG2P2 to support the mono- and di-thio substitutions, along with OG2S1 to model the terminal oxygen connected to the S-P bond.

as well as bug fixes and stability improvements.

2.16 Version 2017.1

This version is the initial release. This version includes the following packages:

- SILCS command line interface
- SILCS-MC command line interface
- SSFEP command line interface
- CGenFF program and parameter version 2.0.0

  CGenFF version 2.0.0 is the second release of CGenFF, extending support to a larger variety of drug-like molecules to be used routinely in computer-aided drug design projects. Specifically, it improves treatment of halogen bonds, by introducing lone-pairs onto the halogen atoms of aromatic systems. Additionally, support is now extended to four-membered oxetane

and glycoluril moieties. This is driven largely by improvements in the newly released version 4.0 of CGenFF force field.

CHAPTER

THREE

SILCSBIO SUITES AND FEATURES

3.1 Overview

SilcsBio offers a range of tools for computer-aided drug design. Our users include academic and government research institues as well as large pharmaceutical companies. Our current offerings are:

- **Small Molecule (SM) Suite**

  The SM Suite provides tools for end-to-end small molecule ligand drug design. Map chemical functional group affinity patterns of your target protein, identify ligand binding sites, virtually screen our databases of up to 1.9 billion compounds with target-based pharmacophore models, dock your ligands of interest, and optimize your lead compound to design your small molecule drug.

- **Covalent Suite**

  The Covalent Suite streamlines your covalent drug design process by combining physics-based chemical functional group affinity patterns and machine learning models. Identify the most reactive residues within your protein, predict the optimal warheads for your target residue, model the binding of your covalent inhibitor, and optimize your covalent inhibitor to maximize its potency.

- **Kinetics Suite**

  The Kinetics Suite provides tools for predicting ligand dissociation kinetics, which is often more relevant to drug efficacy than binding affinity alone. Identify ligand dissociation pathways for your target protein, predict ligand dissociation kinetics, and identify atomic or functional group contributions to unbinding events to optimize the kinetic profile of your drug candidate.

- **SM+ Suite**

  Combine the power of the SM, Covalent and Kinetics Suites with the SM+ Suite. The SM+ Suite provides users with all the features in the SM and Covalent Suites in one package.

- **Biologics Suite**

  The Biologics Suite provides tools that rationalize the selection of excipients for formulating your protein-based therapeutic. Predict regions of your therapeutic protein prone to self-aggregation, map chemical functional group affinity patterns for your protein, predict excipient interaction sites, and characterize excipient-protein interactions in relation to protein-protein aggregation to identify excipients that stabilize the active fold of your protein and prevent aggregation thereby reducing viscosity for your biologic's optimal formulation.

- **SSFEP Suite**

  The SSFEP Suite leverages the single step free energy pertubation (SSFEP) method to circumvent the large computational cost of free energy perturbation (FEP) while maintaining comparible accuracy for small functional group modifications. Introduce functional group modifications to your parent ligand and rapidly evaluate hundreds of potential modifications involving multiple sites on your parent ligand for your target protein.

- **CGenFF Suite**

  The CGenFF Suite enables accurate molecular dynamics (MD) simulations of biomolecular systems. Rapidly generate force field (FF) topologies and parameters for novel ligands on the fly, including covalently bonded ligands, run standard molecular dynamics simulations, and perform structural and energetic analyses or your biomolecular system.

For more details on what features are avalable for each suite, please see *Suite Feature and Module Comparison*.

3.2 Suite Feature and Module Comparison

The suites and their included features and modules are listed in the table below. Features marked with "CLI" are available only in the command line interface, and features marked with "CLI | GUI" are available in both the command line interface (CLI) and the SilcsBio graphical user interface (GUI). If you have any questions about the suites or any particular feature, please contact info@silcsbio.com.

Modules ↓	SilcsBio Software Suites						
	SM (small molecule)	**Covalent**	**Kinetics**	**SM+**	**Biologics**	**SSFEP**	**CGenFF**
SILCS simulations and FragMaps	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI		
SILCS-MC Docking and Pose Refinement	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI		
SILCS-MC Ligand Optimization	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI	CLI + GUI		
SILCS-Hotspots	CLI + GUI		CLI + GUI	CLI + GUI	CLI + GUI		
SILCS-Pharm	CLI + GUI			CLI + GUI			
SILCS-BML (Bayesian Machine Learning)	CLI		CLI	CLI			
SILCS-Water	CLI			CLI			
SILCS-Covalent		CLI		CLI			
SILCS-Pathway			CLI	CLI			
SILCS-Kinetics			CLI	CLI			
Protein - Protein Interaction (PPI)					CLI + GUI		
Formulation Evaluation					CLI + GUI		
Protein Charge Prediction					CLI		
SSFEP						CLI + GUI	
CGenFF							CLI + GUI
CGenFF-Covalent							CLI
CGenFF Parameter Optimizer							CLI
MD Simulations Module							CLI
MD Simulations with Probes							CLI

CHAPTER

FOUR

SILCSBIO INSTALLATION

4.1 Hardware and Software Requirements

4.1.1 Minimum Hardware Requirement

SilcsBio software requires relatively robust computational resources for the molecular dynamics (MD) components of the SILCS and SSFEP protocols. For example, computing SILCS FragMaps for a 35 kDa target protein takes 18-24 hours of walltime when run in parallel on ten GPU nodes, each equipped with 8 3-GHz CPU cores and 1 GPU (NVIDIA GeForce RTX 3080 or later). In the case of SSFEP, walltime using these GPU-equipped nodes will be 3-4 hours.

SilcsBio software is designed to run the compute-intensive MD on a cluster using a cluster queue management system such as OpenPBS, Sun Grid Engine, or SLURM. With both SILCS and SSFEP simulations, subsequent evaluation of relative ligand affinities takes seconds to minutes on a single CPU core, allowing for modifications to be rapidly evaluated for a large number of ligands to a given target.

For customers without ready access to an appropriate in-house computing cluster, SilcsBio offers two possible solutions. The first solution is for SilcsBio to perform computations as a service and supply data to the customer for subsequent in-house analysis. For this service, in the case of performing SILCS simulations, SilcsBio requires only the structure of the target, and, in the case of performing SSFEP simulations, SilcsBio requires only the structure of the protein-parent ligand complex. Depending on the customer's choice of SSFEP or SILCS, no intellectual property disclosure to SilcsBio in the form of proposed chemical modifications to the parent ligand (SSFEP) or even the parent ligand itself (SILCS) is required. The second solution is for SilcsBio to assist customers with setting up their own virtual cluster using Amazon Web Services (AWS). Please contact info@silcsbio.com for additional information on these solutions.

4.1.2 Python 3 Requirement

The SilcsBio server software requires a working Python 3 installation. We recommend using Miniconda (https://docs.conda.io/en/latest/miniconda.html) for installing Python 3. Once Python 3 is installed, you will need to install some additional Python packages.

To do so, run the following commands:

1. Create a new conda environment:

```
conda env create -f $SILCSBIODIR/utils/python/environment.yml
```

2. Activate the new conda environment:

```
conda activate silcsbio
```

   OR export the environment variable:

```
export PATH="<path-to->/miniconda3/envs/silcsbio/bin:$PATH"
```

   Note that the path to the conda environment may vary depending on the installation location. Also, user may want to add the above command to their `.bashrc` file which will automatically activate the environment upon login.

4.1.3 GROMACS Requirement

Both SILCS and SSFEP MD simulations require the GROMACS MD software. We recommend GROMACS version 2025.3. GROMACS can be obtained at https://manual.gromacs.org/2025.3/index.html and must be installed in order to use the SilcsBio software package.

We strongly recommend building GROMACS with GPU acceleration enabled. Below is a sequence of commands for GROMACS v. 2025.3 installation with GPU acceleration:

```
cd <GROMACS source directory>
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=on \
  -DREGRESSIONTEST_DOWNLOAD=on \
  -DGMX_GPU=CUDA \
  -DGMXAPI=off \
  -DBUILD_SHARED_LIBS=off \
  -DGMX_PREFER_STATIC_LIBS=on \
  -DGMX_BUILD_SHARED_EXE=off \
  -DCMAKE_INSTALL_PREFIX=<GROMACS install location>
make
make install
```

If your compute nodes do not have GPUs, use the following commands:

```
cd <GROMACS source directory>
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=on \
  -DREGRESSIONTEST_DOWNLOAD=on \
  -DGMXAPI=off \
  -DBUILD_SHARED_LIBS=off \
  -DGMX_PREFER_STATIC_LIBS=on \
  -DGMX_BUILD_SHARED_EXE=off \
  -DCMAKE_INSTALL_PREFIX=<GROMACS install location>
make
make install
```

Please refer to http://manual.gromacs.org/documentation/current/install-guide/index.html for further details.

4.2 Installation

4.2.1 Server Software Installation

The SilcsBio server software package is delivered as a zip-compressed file. Unzip and place the files to an accessible location. The files have the following directory structure:

```
silcsbio.$VERSION/
  cgenff/
  cgenff-covalent/
  cgenff-optimizer/
  data/
  examples/
  lib/
  lsqfit/
  md/
  ppi/
  programs/
  silcs/
  silcs-biologics/
  silcs-bml/
  silcs-hotspots/
  silcs-mc/
  silcs-memb/
```

(continues on next page)

(continued from previous page)

```
  silcs-pharm/
  silcs-rna/
  ssfep/
  ssfep-memb/
  templates/
  utils/
  VERSION
```

The top-level `silcsbio.$VERSION/` folder contains software for running SILCS and SSFEP simulations. The `programs/`, `silcs*/`, and `ssfep*/` folders contain executable code.

The `templates/` folder contains templates for job handling and input scripts. **Note that for academic users, some template files** (`.tmpl`) **may need to be edited with information for your queuing system.**

If you are a system administrator, place the top-level `silcsbio.$VERSION/` folder where it can be accessed by other users, such as `/opt/silcsbio/`. If you are a single user, you may place the folder in your home directory.

For SilcsBio server software to work, the two shell environment variables `GMXDIR` and `SILCSBIODIR` need to be set. To do so, replace <gromacs/bin> and <silcsbio> with the complete file paths for the corresponding folders:

```
# bash
export GMXDIR=<gromacs/bin>
export SILCSBIODIR=<silcsbio>

# python environment if not already set
export PATH="<path-to->/miniconda3/envs/silcsbio/bin:$PATH"
```

Currently, the SilcsBio server software is compatible only with the Bash shell environment. You may insert the above environment variable settings in `.bashrc` for convenience.

4.2.2 Graphical User Interface (GUI) Installation

The SilcsBio Graphical User Interface (GUI) enables the user to run SILCS and SSFEP simulations and analyze results through a GUI instead of the command line. The SilcsBio GUI is available for Windows, macOS, and Linux. Please download and install the software on your local desktop or laptop computer.

In addition to providing standalone features such as FragMap visualization and ligand modification, the SilcsBio GUI can set up, launch, manage, and analyze compute-intensive SILCS and SSFEP simulations. Enabling this functionality requires a simple configuration step to allow the GUI to communicate with your SilcsBio server software.

Please follow the *Remote Server Setup* process as described in *Graphical User Interface (GUI) Quickstart*. Contact support@silcsbio.com if you need help with this process.

4.2.3 Amazon Web Services (AWS) Cluster Setup

To configure the AWS ParallelCluster, please follow the instructions provided by AWS, under the "Slurm" tab, in the following link:

https://docs.aws.amazon.com/parallelcluster/latest/ug/install-v3-configuring.html

Please note AWS' terminology for partition and queue. For AWS, a partition is a group of AWS Region and Service objects. The AWS partition, which is based on the region determines what services are available. For AWS, a queue is what traditionally (i.e., in Slurm terminology) would be called a partition, which can be considered job queues, each with constraints including, but not limited to, the number of compute resources in the queue or maximum number of running jobs. Additional information on the Slurm workload manager is available at https://slurm.schedmd.com/documentation.html. For the purpose of this section and consistency with AWS instructions, we will follow the terminology of AWS.

As detailed in the AWS ParallelCluster configuration instructions linked above, using the `pcluster configure` CLI command will initiate prompts for you to enter information required to configure your AWS cluster.

When prompted to choose the job scheduler, enter slurm.

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]: slurm
```

When prompted to choose the AWS queue configuration, enter the following setup:

```
Number of queues: 2
Name of queue 1: gpu
Number of compute resources for gpu: 10
Compute instance type for compute resource 1 in gpu: g5.2xlarge
Name of queue 2: cpu
Number of compute resources for cpu: 10
Compute instance type for compute resource 1 in cpu: c5.2xlarge
```

Note: Please enter an integer value for `Number of compute resources`. We recommend multiples of 10 (e.g., if you plan on simultaneously running SILCS simulations for 5 proteins, then the entry for `Number of compute resources` would be `50`.)

Contact support@silcsbio.com if you need help with this process.

4.2.4 PHARMER Installation

Please refer to https://github.com/SilcsBio/pharmer

CHAPTER

FIVE

QUICKSTART

5.1 Graphical User Interface (GUI) Quickstart

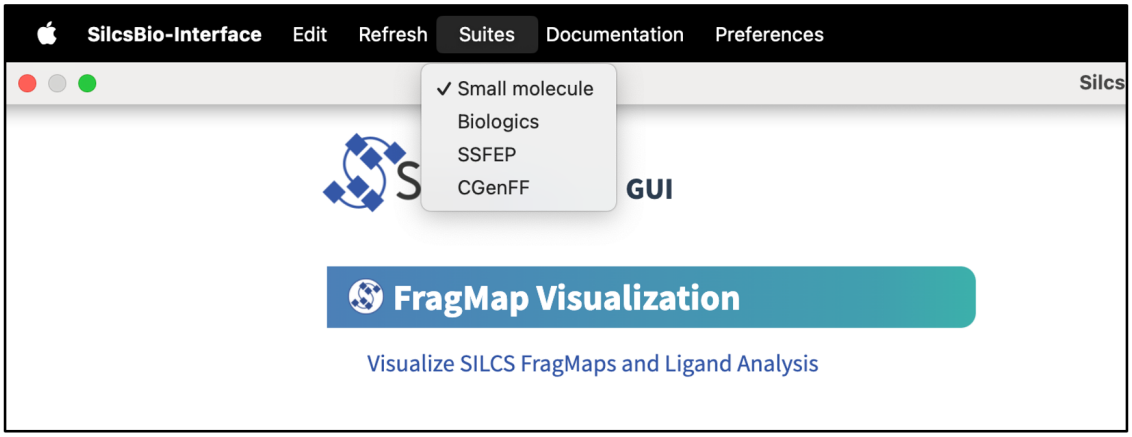
This chapter provides a step-by-step introduction on how to use the SilcsBio Graphical User Interface (GUI).

5.1.1 SilcsBio GUI Suite Selection

The SilcsBio GUI provides convenient access to cutting-edge SILCS and CGenFF technologies for computer-aided drug discovery. Currently, four suites are available through the SilcsBio GUI:

1. Small Molecule Suite
2. Biologics Suite
3. SSFEP Suite
4. CGenFF Suite

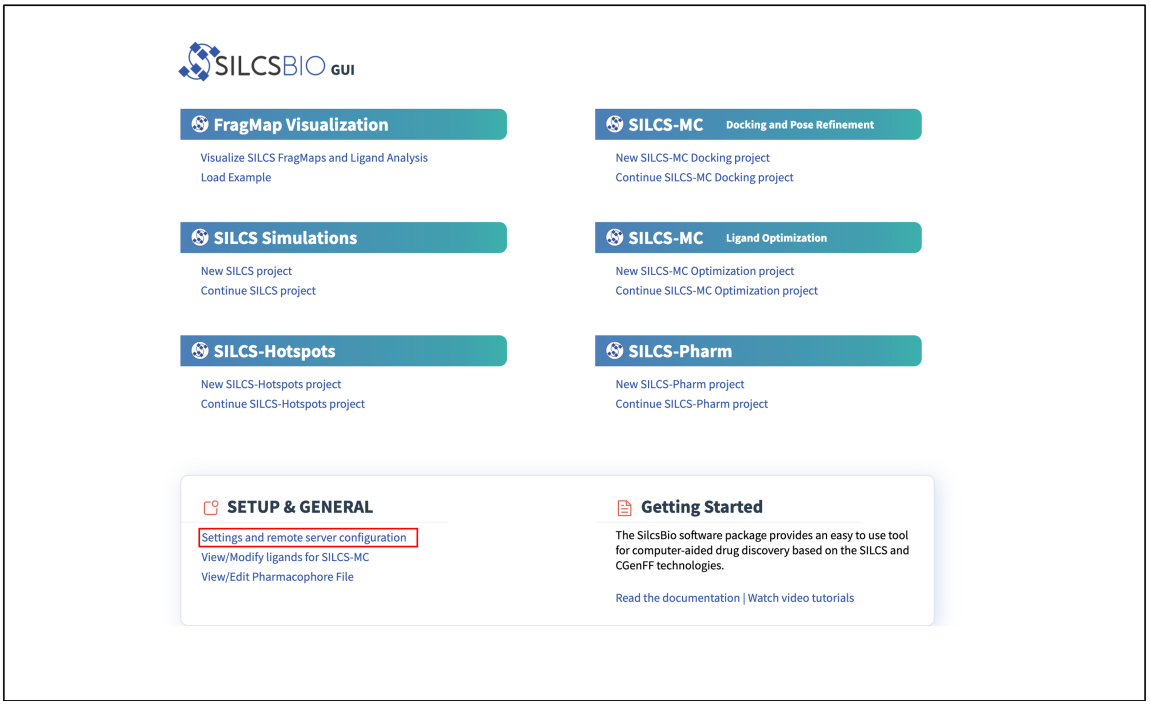
Upon opening the SilcsBio GUI, the user will directly enter the Small Molecule Suite. To access different suites, select the desired suite from the menu bar.



Additional information on the capabilities of each suite are overviewed in the current chapter as well as detailed in the following chapters of this documentation.

5.1.2 Remote Server Setup

The SilcsBio GUI is designed to work with the server installation of the SilcsBio software. Launch the GUI and select *Settings and remote server configuration*.

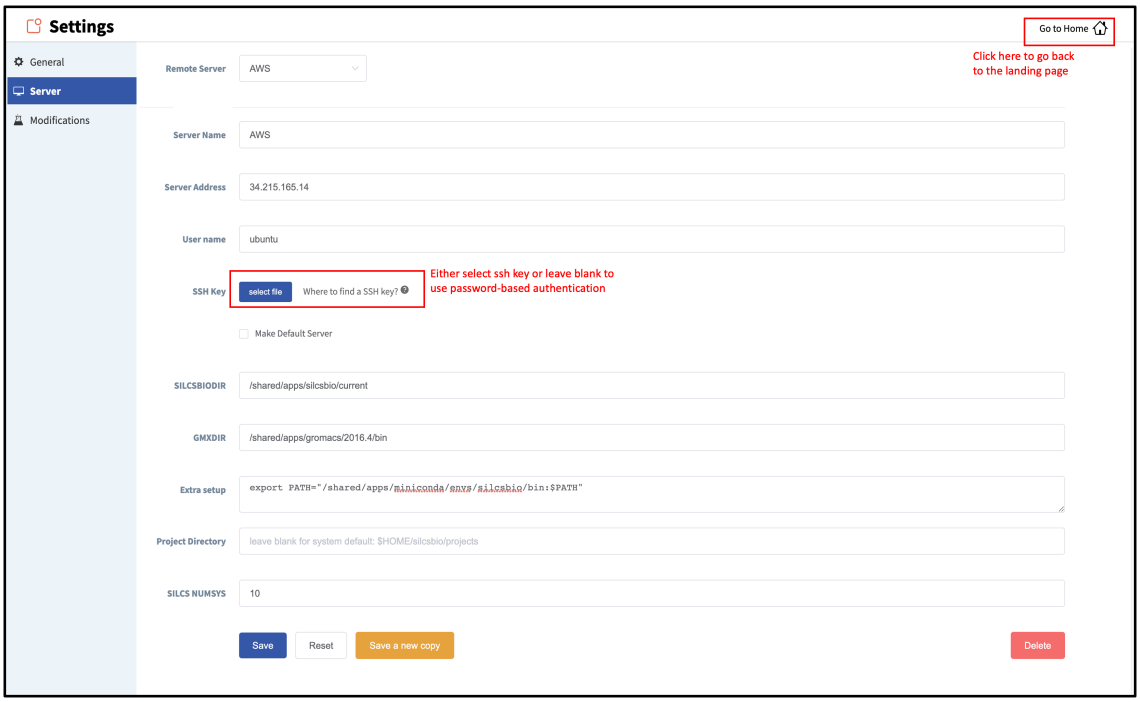


Within the "Settings and remote server configuration" page, select *Server* and enter the remote server information, such as Server Address (IP address), Username, and an SSH key to the server. If you do not have an SSH key to the server, leave it blank. The GUI will ask you the password to the remote server instead. Select the "Make Default Server" checkbox if you would like to set this server as your default server. This server will be selected as a default in other parts of the GUI.

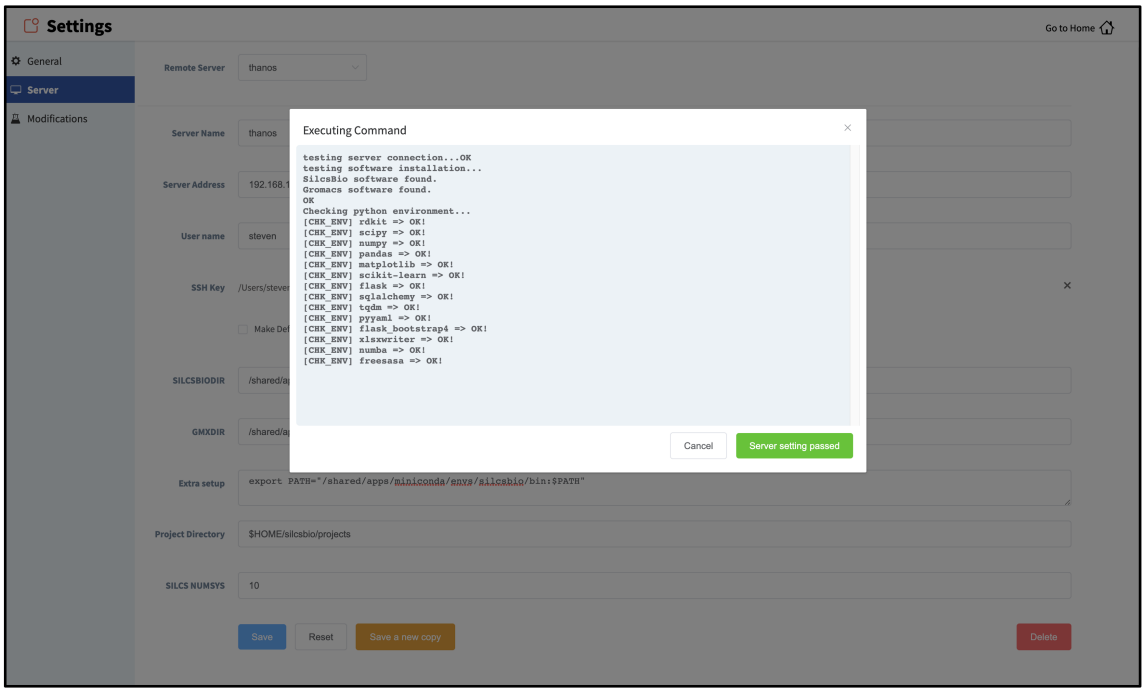
You will also need to enter `SILCSBIODIR` and `GMXDIR` file path values. These should match the values on your server. You may use the "Extra setup" field to pass additional commands to the server, such as exporting environment variables or loading modulefiles.

Note: The SilcsBio GUI will enter a reset environment on the remote server. Thus, you need to manually add all necessary packages (e.g., python) in the PATH.

You may use the "Project Directory" field to define the directory on your server where server job input files will be created and server job outputs stored. Typical SILCS simulations produce output files in excess of 100 GB, so please select a project directory file folder with appropriate storage capacity. The "SILCS NUMSYS" field determines how many SILCS jobs are launched for creating FragMaps. Set this to an even integer; we recommend "10" to maximize convergence or "8" to minimize resource use.



Once all information is entered, click the "Save" button. The GUI will save your entries and confirm that you have a working connection to the server.



Please contact support@silcsbio.com if you need additional assistance.

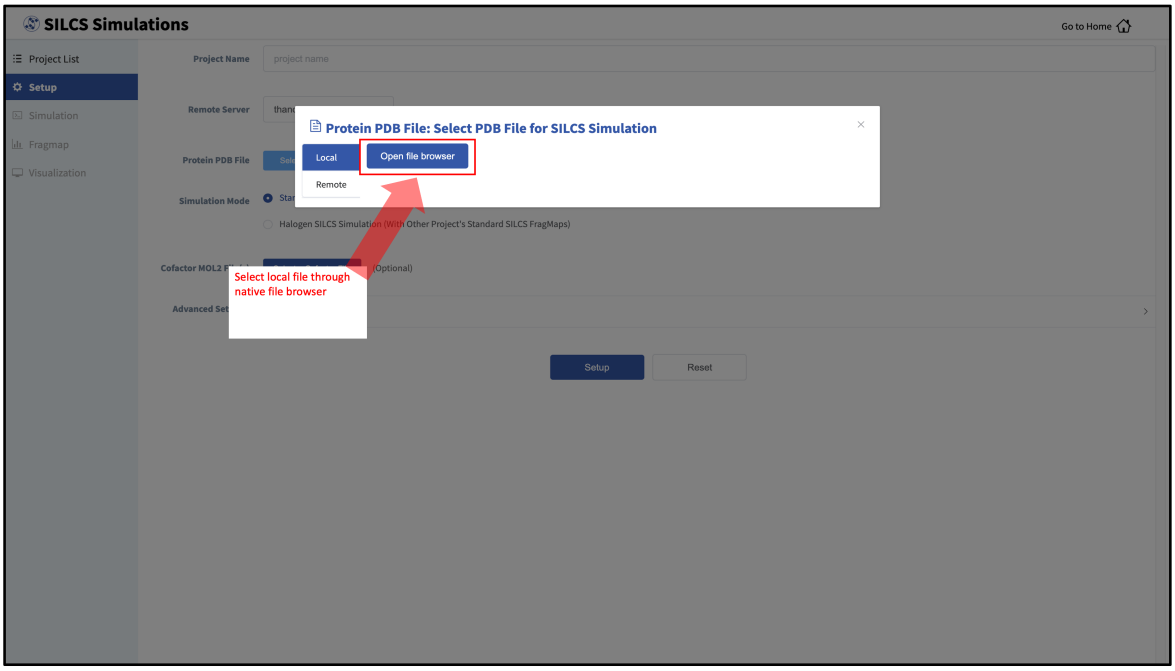
5.1.3 File and Directory Selection

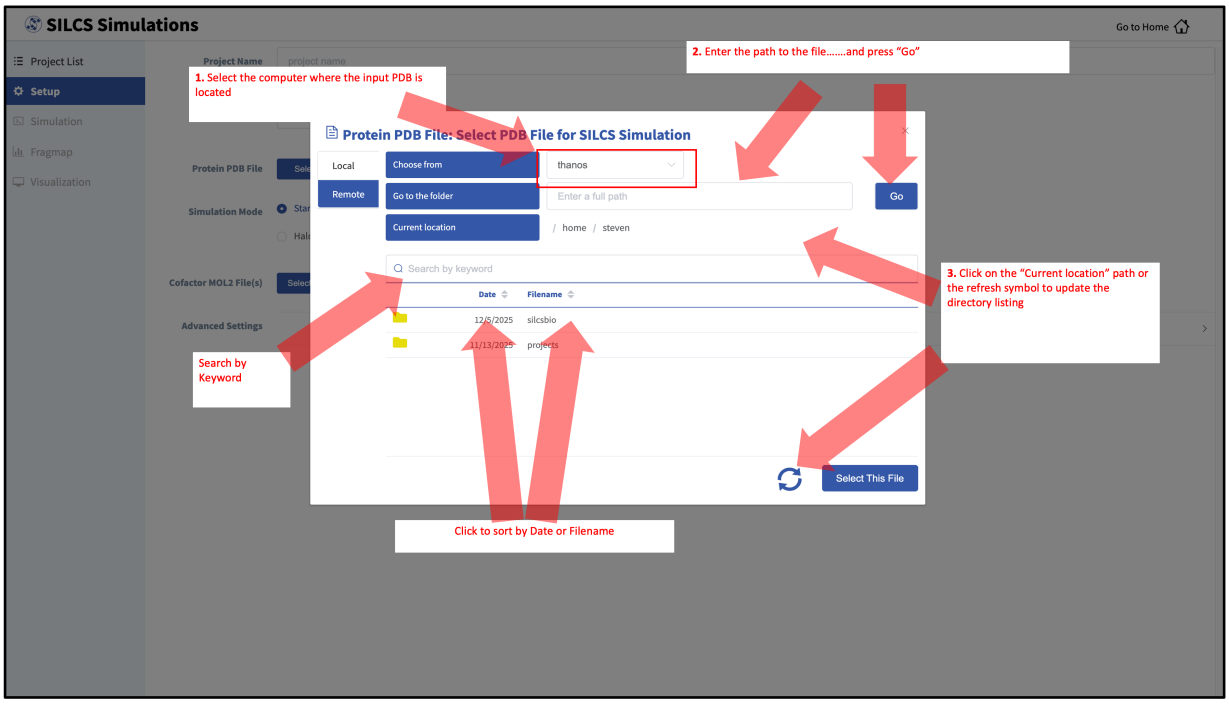
To run the SilcsBio software, you will need to provide protein and/or ligand input files. The SilcsBio GUI has a standardized user interface that allows you to choose these input files from either the computer on which you are running the GUI or from a remote server you have previously configured. Steps to do this, illustrated in the below figures, are as follows:

1. Choose the server on which is the physical location of the input file is located. If you would like to load a file located on the computer on which you are running the GUI, click *Local*; otherwise, click *Remote* and select the remote server name of a server you have previously configured through the *Remote Server Setup* process.

2. If the file is located on the local computer, click *Open file browser*. This will open your local computer's native file browser. Select your file following the conventions of your local computer.

   Otherwise, if the file is located on a remote server, type in the folder where the file is located using the "Go to the folder" field and then click the associated *Go* button. This will update the "Current location" to this folder. Click on the last portion of the "Current location" path (e.g. "silcsbio" in "/state/partition1/home/asukaorr/silsbio" in the below example) to update the directory listing located under that path. If you click on any other portion of the "Current location" path, you will be taken to that folder. Double clicking on a folder in the directory listing will move you into that folder. In the directory listing, click on a file to highlight it and click the *Select This File* button at the bottom to finish your file selection.



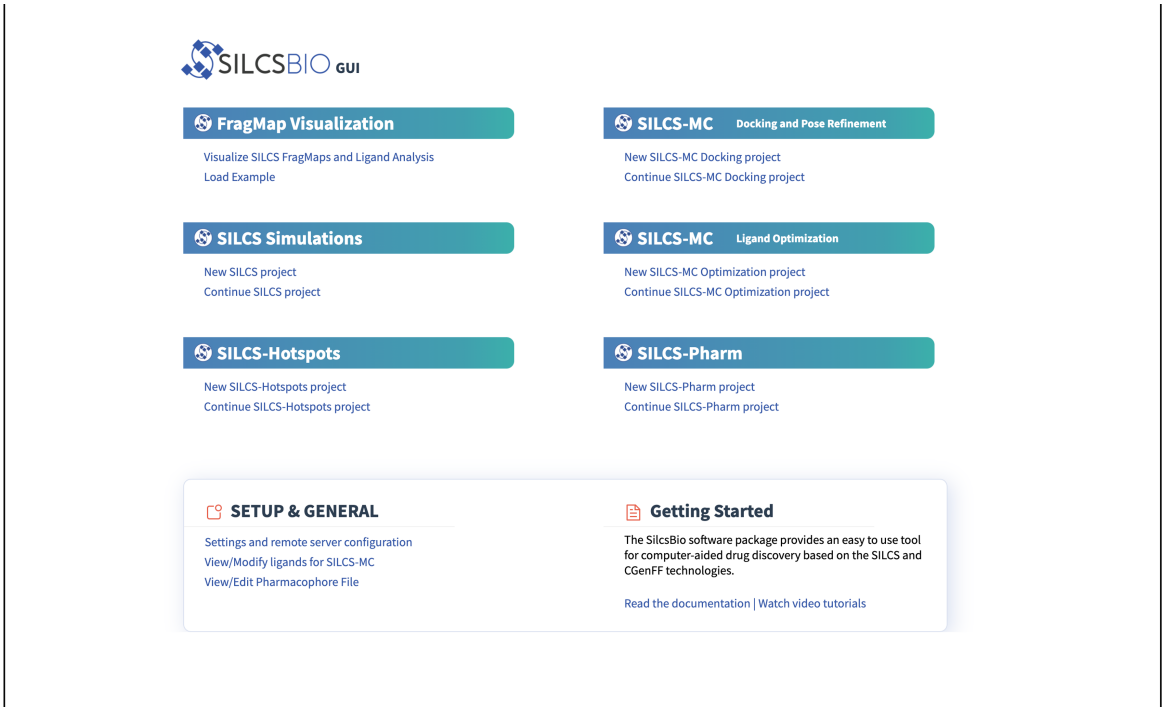


Follow a similar process to select a directory for those tasks requiring directory selection.

Tip: For those tasks needing FragMaps as inputs, "FragMap Location" needs to be a directory. The SilcsBio GUI assumes a directory path of the form <basename>/maps/. Select <basename> for your "FragMap Location". The GUI will automatically look for the maps/ subdirectory and load FragMaps from that subdirectory. It will also automatically load the protein pdb file if one is located in the <basename> directory. It may take a few seconds for the GUI to download your input FragMaps if they are not on localhost.

5.1.4 Small Molecule Suite Quickstart

When the SilcsBio GUI is opened, the user will directly enter the Small Molecule Suite. The Home page of the Small Molecule Suite features a number of SILCS technologies.



The SILCS technologies can be combined to follow a logical order for computer-aided drug discovery/design:

1. Target proteins are characterized by SILCS FragMaps using *SILCS Simulations*.
2. Potential binding sites are identified using *SILCS-Hotspots*.
3. Pharmacophore models are built at identified binding sites using *SILCS-Pharmacophore*.
4. Potential drugs are docked and evaluated with *SILCS-MC Docking and Pose Refinement*.
5. Lead compounds are modified and optimized using *SILCS-MC Ligand Optimization*.

All SILCS technologies revolve around FragMaps generated from SILCS simulations. A quick-start to visualize SILCS FragMaps (*SILCS FragMap Visualization Quickstart*) and to perform the SILCS simulations needed to generate SILCS FragMaps is provided in this chapter (*SILCS Simulations Using the GUI*) with additional details on SILCS simulations provided in *SILCS Simulations*. Information on the background and usage of SILCS-Hotspots, SILCS-Pharmacophore, SILCS-MC Docking and Pose Refinement, and SILCS-MC Ligand Optimization are provided in *SILCS-Hotspots*, *SILCS-Pharm*, *SILCS-MC: Docking and Pose Refinement*, and *SILCS-MC: Ligand Optimization*, respectively.

The Small Molecule Suite also provides convenient access to general tools under *SETUP & GENERAL* at the bottom of the page:

- *Settings and remote server configuration*
- *Modify ligand for SILCS-MC*
- *View/Edit Pharmacophore File*

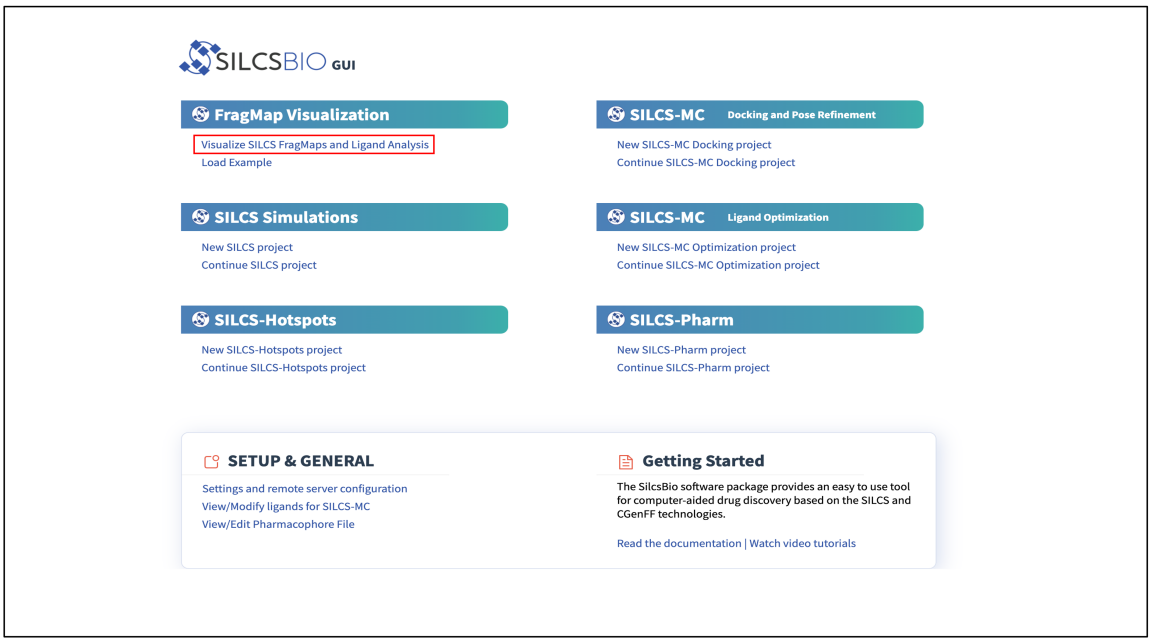
For additional information on these tools, please click on their links.

SILCS FragMap Visualization Quickstart

The SilcsBio GUI provides a convenient platform to visualize FragMaps, as described below. FragMaps can also be visualized using MOE, PyMOL, and VMD as described in *Visualizing SILCS FragMaps*. The following instructions outlines the features in the SilcsBio GUI tool for visualizeing FragMaps:

1. **Open the FragMaps visualization tool:**

   The FragMaps visualization tool can be accessed by clicking *Visualize SILCS FragMaps and Ligand Analysis* at the bottom of the Home page of the SilcsBio GUI:

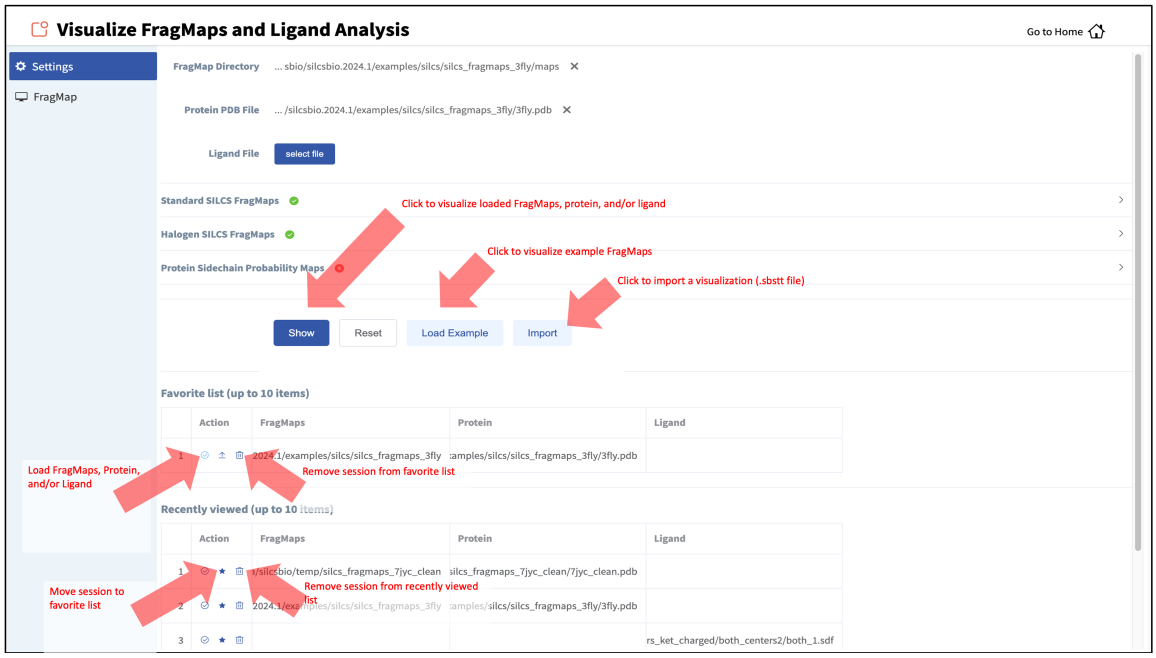


   If you are loading SILCS FragMaps from the Home page, you will be asked to select your FragMap directory (see *File and Directory Selection*). This directory has a standard name, `silcs_fragmaps_<protein PDB>`, where `<protein PDB>` is the name of the input PDB file. This directory was created on the server where you ran the SILCS jobs either when you clicked the "Generate FragMap" button or when you ran the `${SILCSBIODIR}/silcs/2c_fragmap prot=<protein PDB>` command.

   In the case the SilcsBio GUI was used to run the SILCS simulations (see *SILCS Simulations Using the GUI*), once the simulations are complete, the GUI will prompt you to "Generate FragMap" and automatically create FragMaps from the SILCS simulations, download them onto the local computer, and load them, along with the input PDB file, for visualization.

   Alternatively, previously viewed visualization sessions may also be loaded by clicking on the checkmark next to the desired session under "Action". The ten most recently viewed visualization sessions will be listed under "Recently viewed". Frequently viewed sessions

can be saved under the "Favorite list" by clicking on the star icon next to the session under "Action".

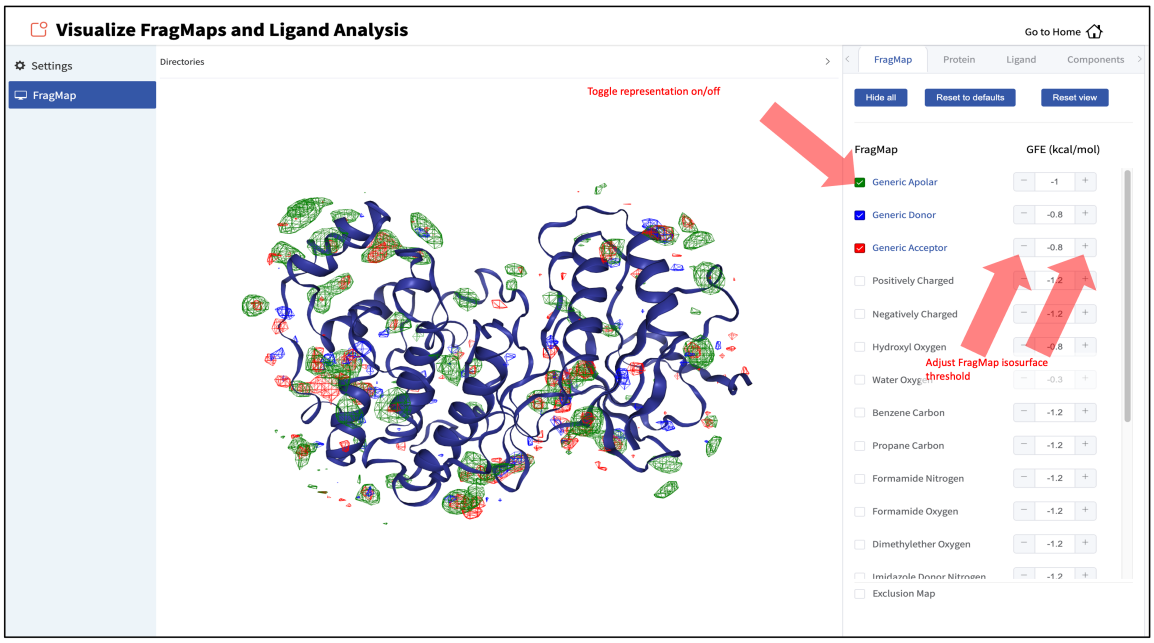


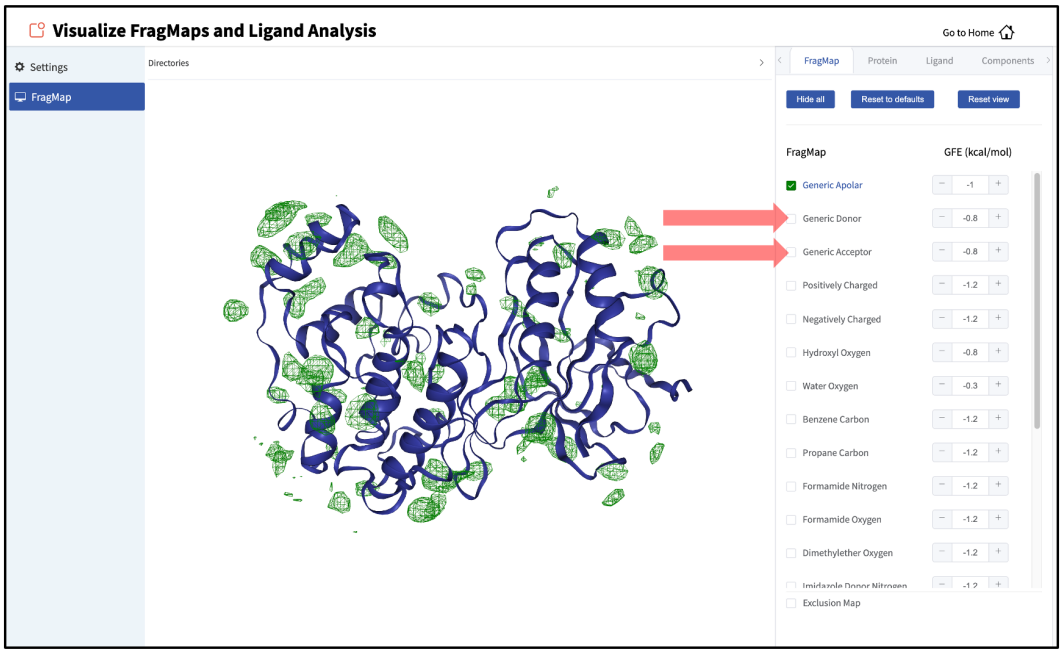
Note: In addition to the FragMaps, the `silcs_fragmaps_<protein PDB>` directory contains the PDB file used to run the SILCS simulations. The SilcsBio GUI will automatically detect and load this PDB file. Additionally, if you have run Halogen SILCS, this directory will also contain the Halogen SILCS FragMaps, which will be automatically detected and loaded by the GUI.

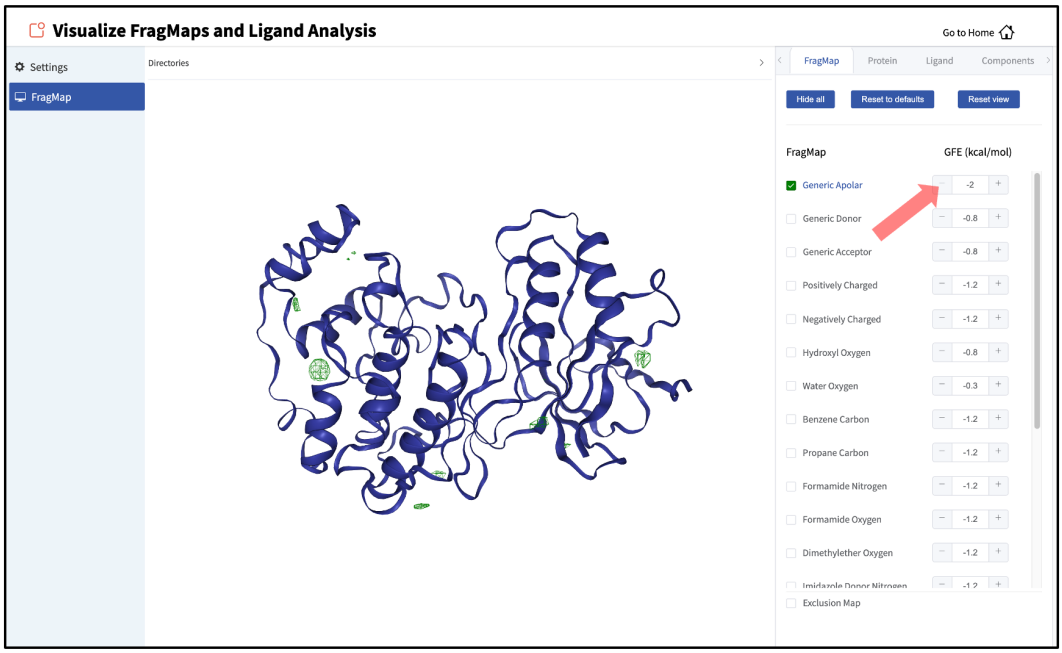
---

2. **Toggle FragMap and protein representations on/off**:

   Specific FragMaps can be hidden or shown by toggling the representations on or off. The FragMap isosurface thresholds can also be adjusted to specific GFE values. The input protein may also be shown in surface and cartoon representation or hidden by toggling the protein representations on or off. Additional details on what each FragMap represents can be found in *Visualizing SILCS FragMaps*.

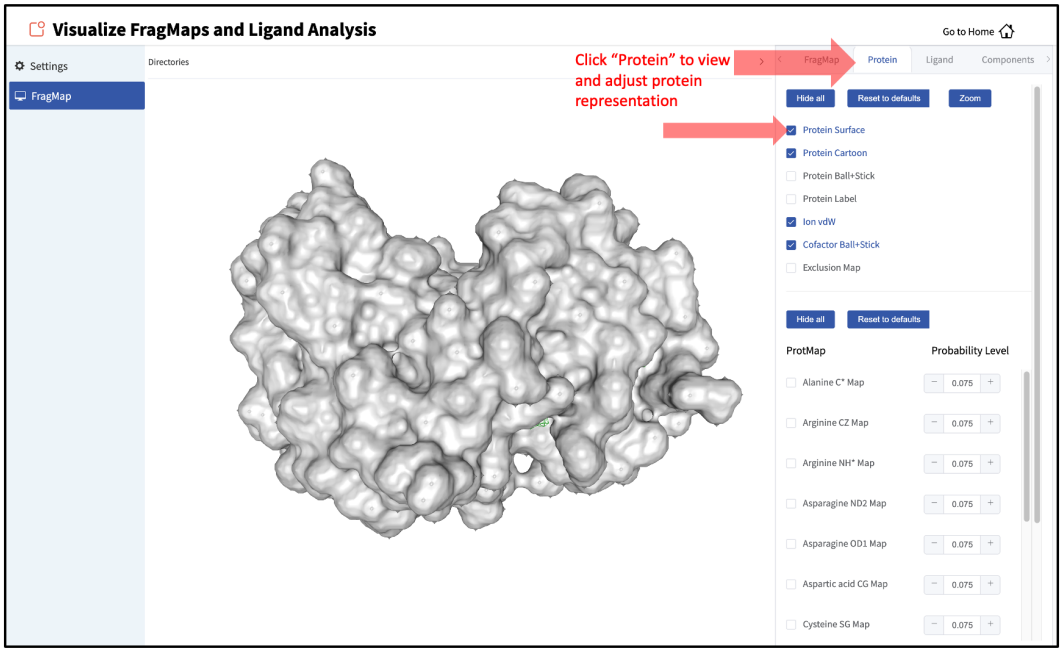

Visualize FragMaps and Ligand Analysis
Go to Home
Settings
FragMap
Directories
Toggle representation on/off
FragMap
Protein
Ligand
Components
Hide all
Reset to defaults
Reset view
FragMap
GFE (kcal/mol)
Generic Apolar
Generic Donor
Generic Acceptor
Positively Charged
Negatively Charged
Hydroxyl Oxygen
Water Oxygen
Benzene Carbon
Propane Carbon
Formamide Nitrogen
Formamide Oxygen
Dimethylether Oxygen
Exclusion Map
Adjust FragMap isosurface threshold


Visualize FragMaps and Ligand Analysis
Go to Home
Settings
FragMap
Directories
FragMap
Protein
Ligand
Components
Hide all
Reset to defaults
Reset view
FragMap
GFE (kcal/mol)
Generic Apolar
Generic Donor
Generic Acceptor
Positively Charged
Negatively Charged
Hydroxyl Oxygen
Water Oxygen
Benzene Carbon
Propane Carbon
Formamide Nitrogen
Formamide Oxygen
Dimethylether Oxygen
Exclusion Map



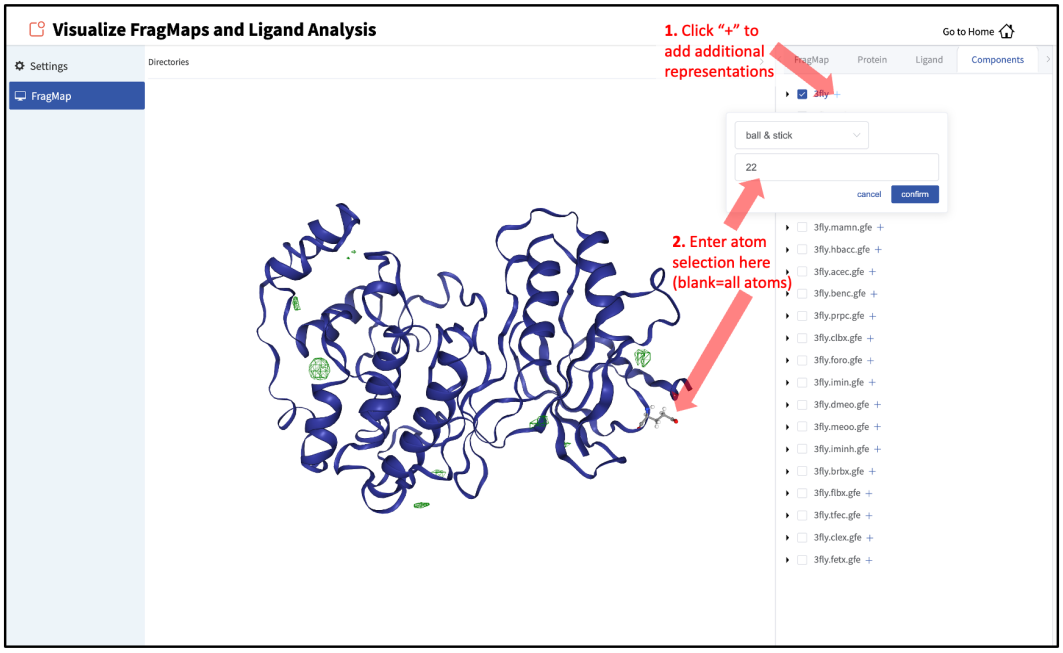
3. **Adjust protein representation**:

   By default, the protein will be shown in cartoon representation. The representation of the protein can be adjusted by clicking the "Protein" tab of the sidebar menu and toggling on or off the desired protein representations. The default protein representation can be restored by clicking on the *Reset to defaults* button.



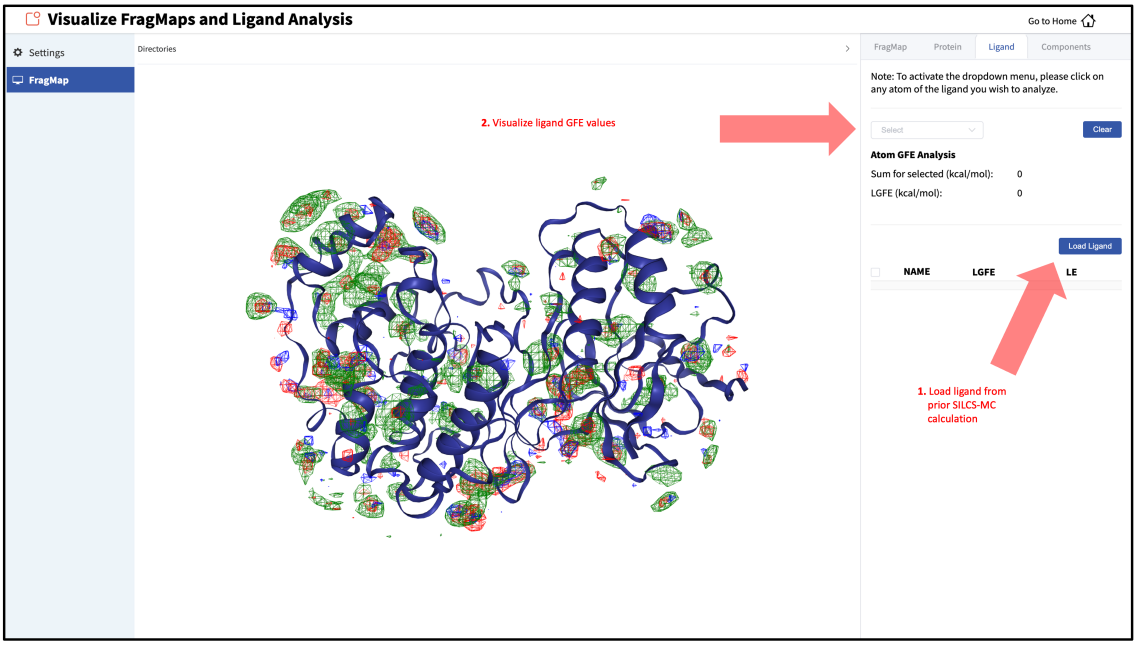
When adding an additional representation, the default is to apply it to all atoms. To apply a representation to a specific set of atoms, click on the "Components" tab of the sidebar menu, click on the "+" adjacent to the protein structure name, and type the atom selection in the

blank box above the "cancel" and "confirm" buttons. The selection syntax is the same as for the NGL molecular structure viewer (see *Atom Selection Syntax*).



4. **Load ligands for visualization**:

   It is also possible to load ligands for visualization. If the ligand file was the output of a SILCS-MC calculation, GFE values for the ligand atoms will be embedded within that file. These values can be visualized and their sum automatically computed by first loading the ligand file, and then using the "Ligand" tab as detailed at the end of the section *SILCS-MC Ligand Optimization Using the SilcsBio GUI*.



SILCS Simulations Using the GUI

To begin a new SILCS project, follow these steps:

1. **Begin a new SILCS project:**

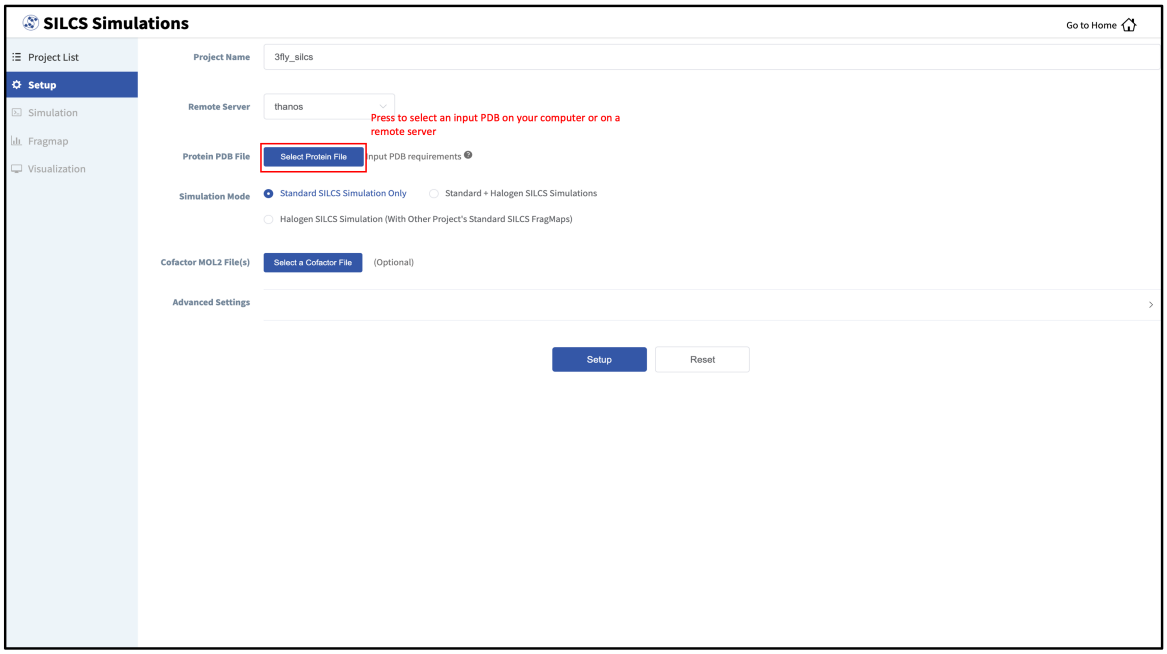
   Select *New SILCS project* from the Home page.

2. **Select the macromolecule target type:**

   Click on the desired macromolecule type (*Globular Protein*, *Membrane Protein*, or *RNA*). For the purposes of this GUI quickstart, the following instructions are focused on running standard SILCS simulations for *Globular Protein* target types. For additional details on SILCS for *Membrane Protein*, or *RNA* target type, please see *SILCS Simulations*.

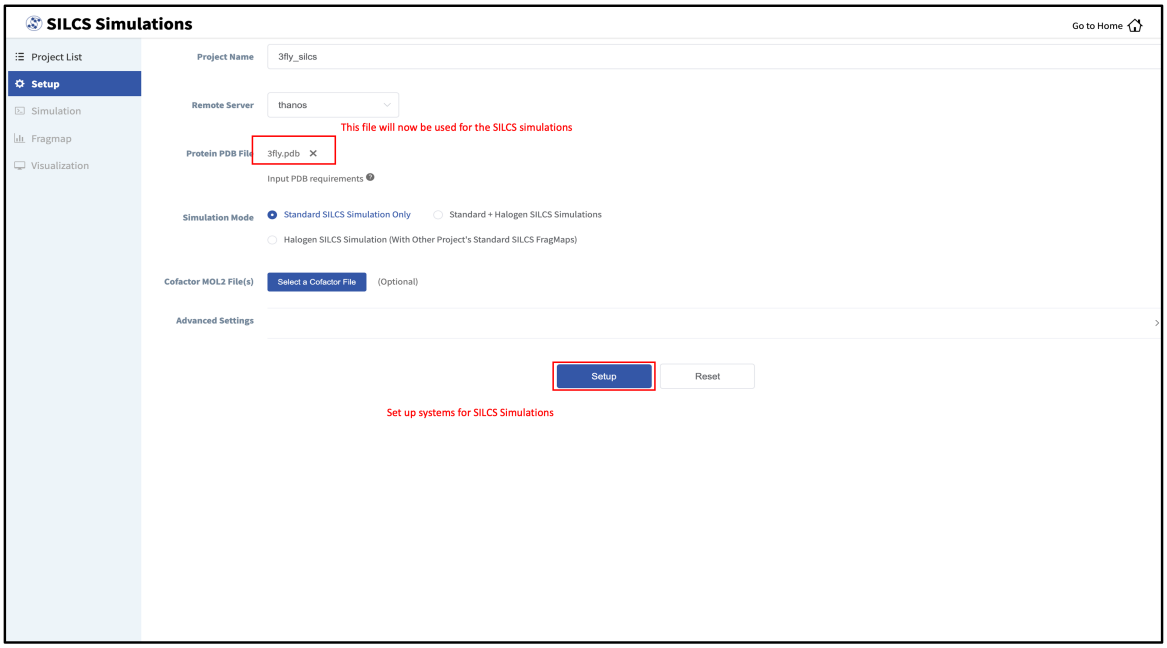
3. **Enter a project name, select the remote server, and upload input files:**

   Enter a project name and select the remote server where the SILCS jobs will run. Input and output files from the SILCS jobs will be stored on this server based on your choice of "Project Directory" during the *Remote Server Setup* process.



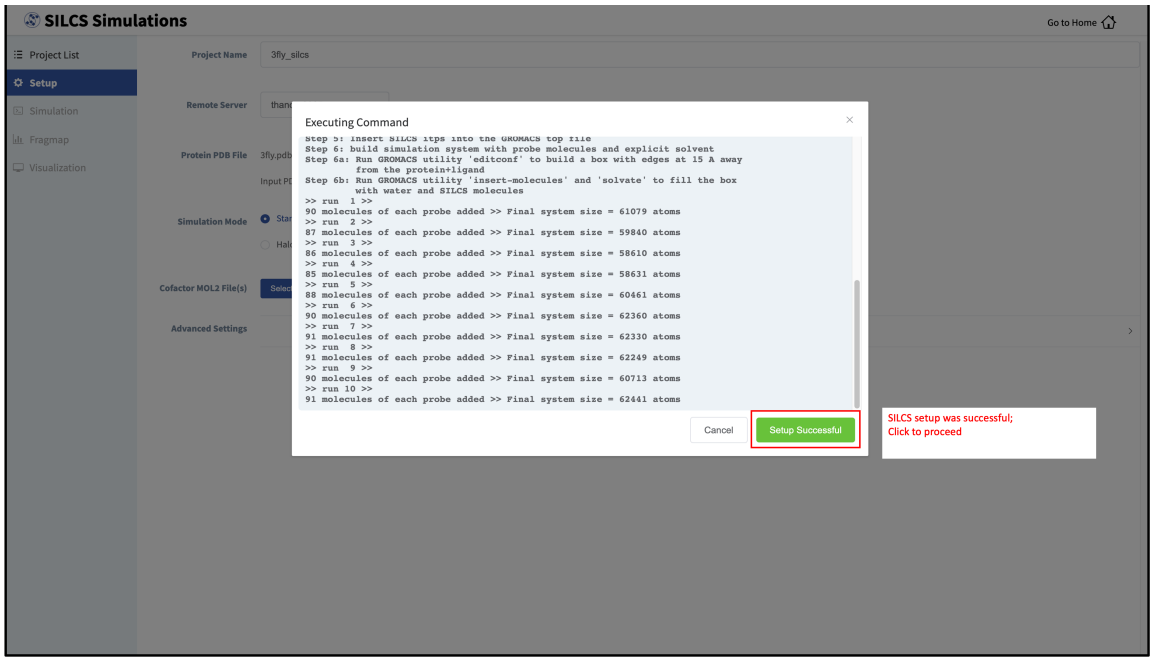
Next, select a protein PDB file. As described in *File and Directory Selection*, choose a file from your local computer ("localhost") or from any server you had configured through the *Remote Server Setup* process. We recommend cleaning your input PDB file before use, including keeping only those protein chains that are necessary for the simulation, removing all unnecessary ligands, renaming non-standard residues, filling in missing atomic positions, and, if desired, modeling in missing loops.

If the GUI detects missing non-hydrogen atoms, non-standard residue names, or non-contiguous residue numbering, it will inform the user and provide a button labeled "Fix?". If this button is clicked, a new PDB file with these problems fixed and with the suffix `_fixed` added to the PDB file base name will be created and used. The SilcsBio GUI will not model residues that are completely missing from the PDB or loops.



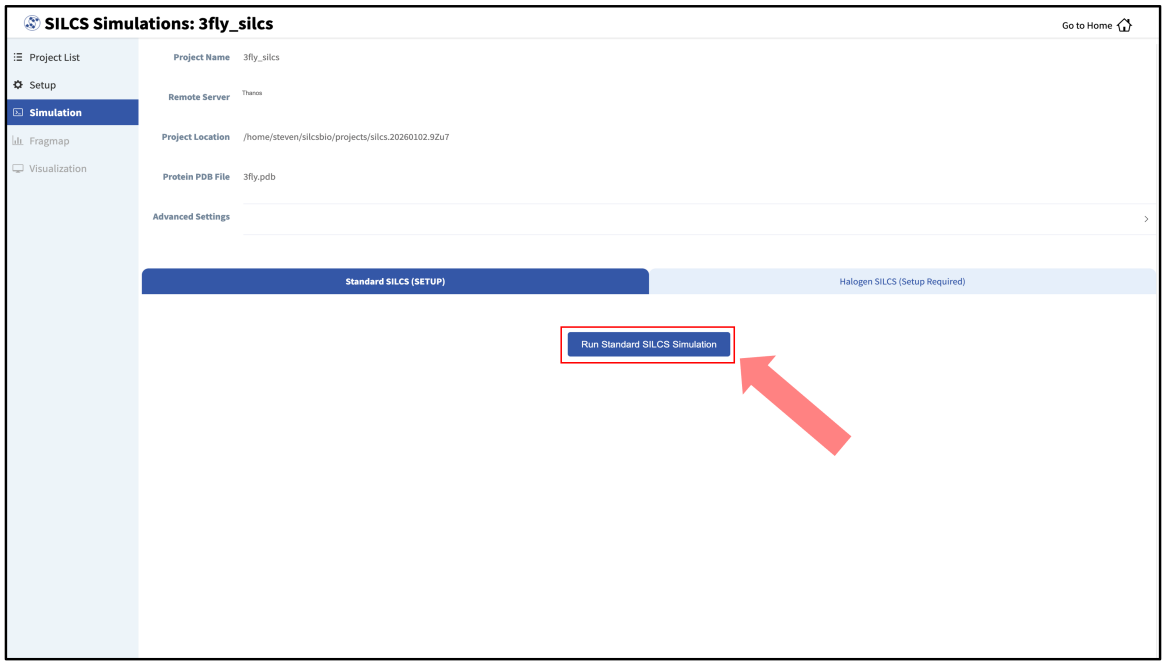
4. **Set up SILCS simulations:**

   Press the "Setup" button at the bottom of the page. The GUI will contact the remote server and perform the SILCS GCMC/MD setup process. During setup, the program automatically performs several steps: building the topology of the simulation system, creating metal-protein bonds if metal ions are found, rotating side chain orientations to enhance sampling, and putting probe molecules around the protein. To complete the entire process may take up to 10 minutes depending on the system size. A green "Setup Successful" button will appear once the process has successfully completed. Press this button to go to the next step.

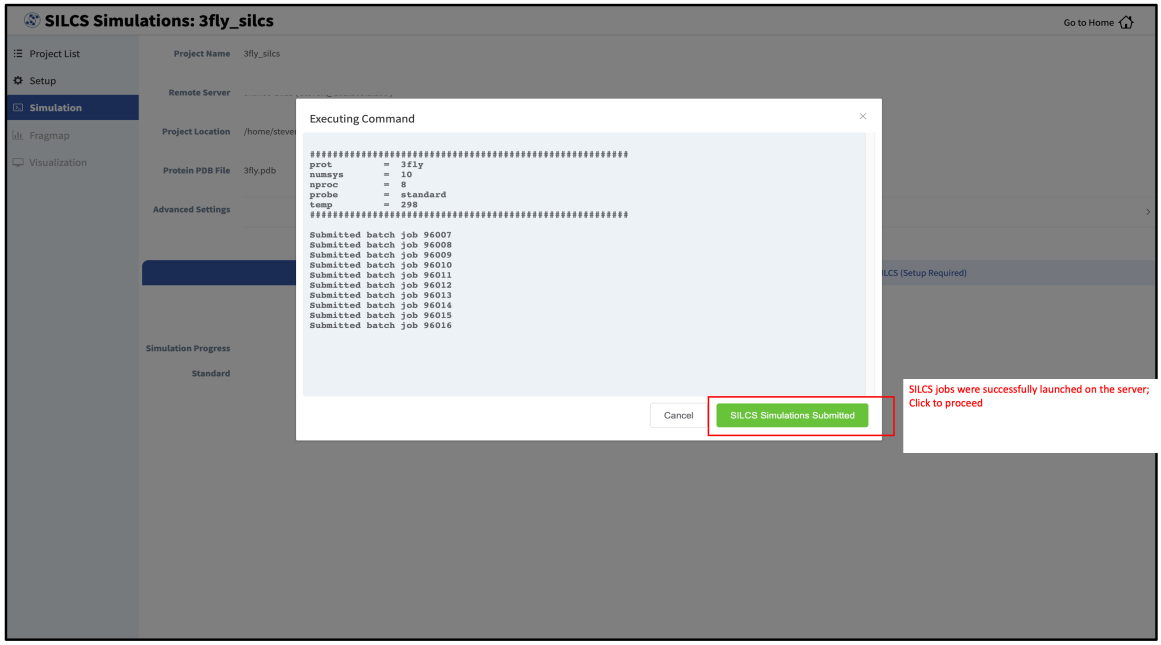


5. **Launch SILCS simulations:**

   Your SILCS GCMC/MD simulation can now be started by clicking the "Run SILCS Simulation" button. Before running, you may wish to double check that you have chosen the desired file folder on the remote server and that it has 100+ GB of storage space.



   Compute jobs will be submitted to the queueing system on the server.



   The status of each job is shown next to its progress bar: "Q" for queued, an orange clock and "Running" for running, "NA" for not submitted, and "F" for failed. When a job successfully

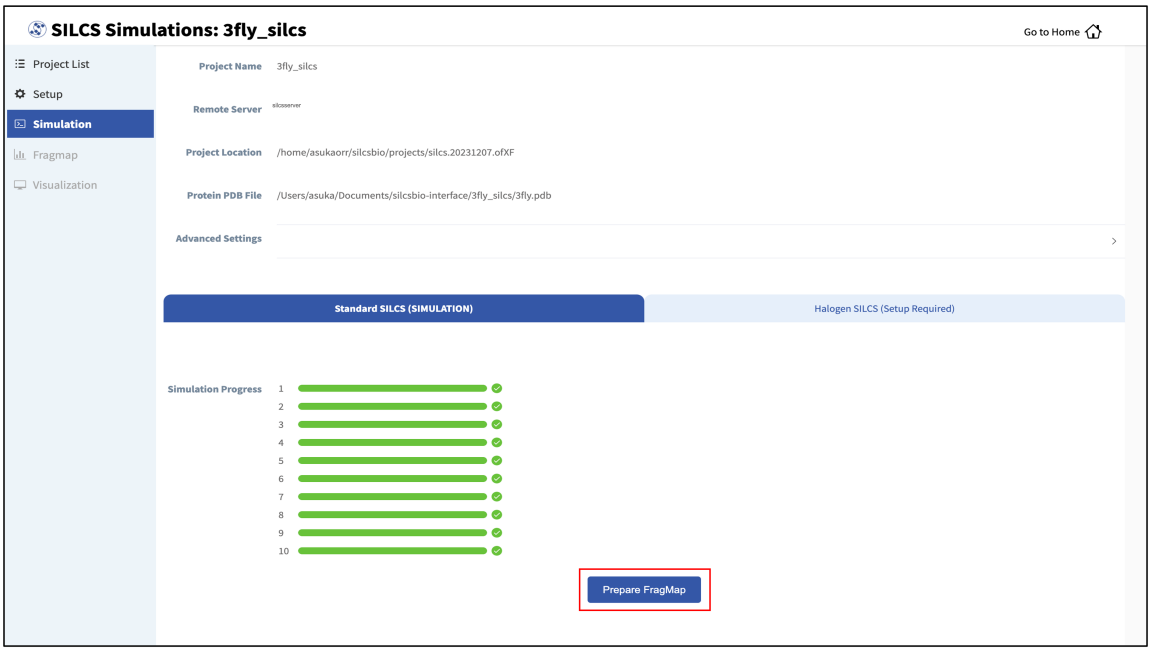
completes, its progress bar will turn green.

Once your jobs are in progress, in either the queued or running state, you may safely quit the SilcsBio GUI or go back to the Home page to do other tasks.

If a job encounters an error or you cancel it before it finishes, its progress bar will turn red. “Restart” will appear next to the progress bar. Pressing “Restart” will resubmit the job to the queue and continue from the last cycle of the calculation so that previous progress on that job is not lost.

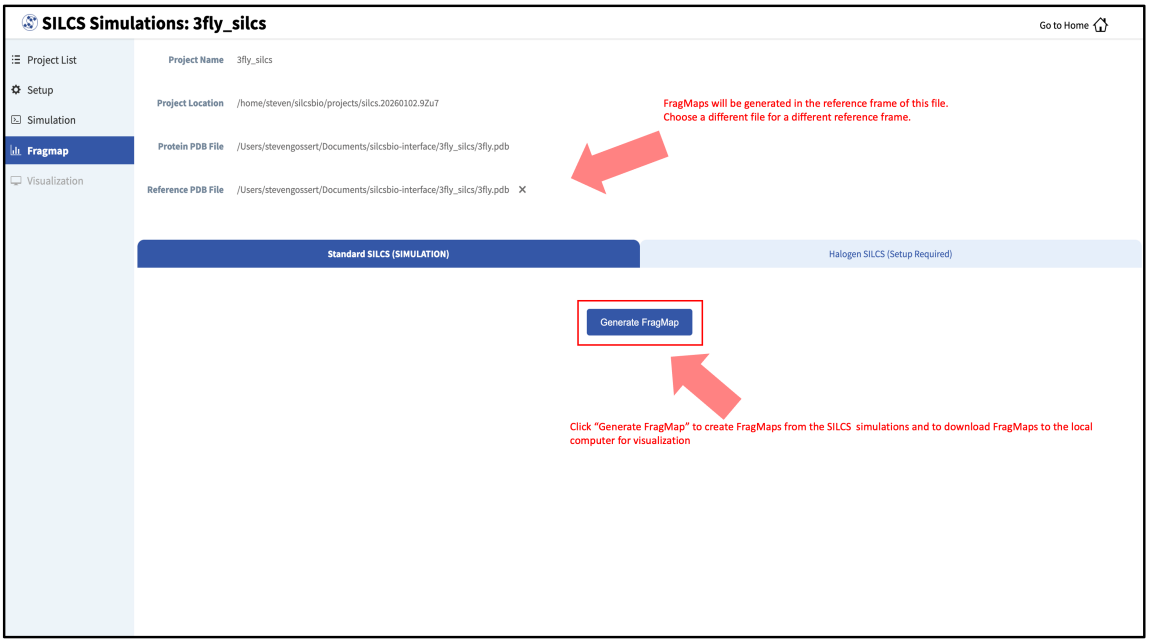
6. **Generate FragMaps:**

   Once your SILCS compute jobs are finished, the GUI can be used to create FragMaps and visualize them. Green progress bars indicate successful job completion. Once all progress bars are green, the “Prepare FragMap” button will appear.

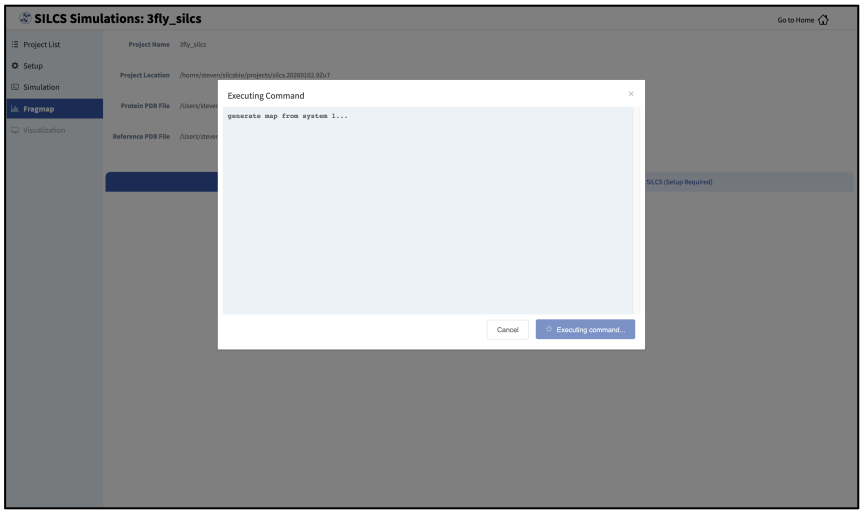


Press this button, and on the next screen, confirm your "Reference PDB File." FragMaps will be created in the coordinate reference frame of this file. Generally, this reference PDB file is the same as the protein PDB file. A different file with the protein in a different orientation can be selected if you want to generate FragMaps relative to that orientation. Click "Generate FragMap" to generate FragMaps and download them to the local computer for visualization.

Tip: If you plan to compare FragMaps from two different protein structures, you will want to generate them with the same orientation. In that case, pre-align your input structures with each other and use these aligned coordinates as your Reference PDB Files.



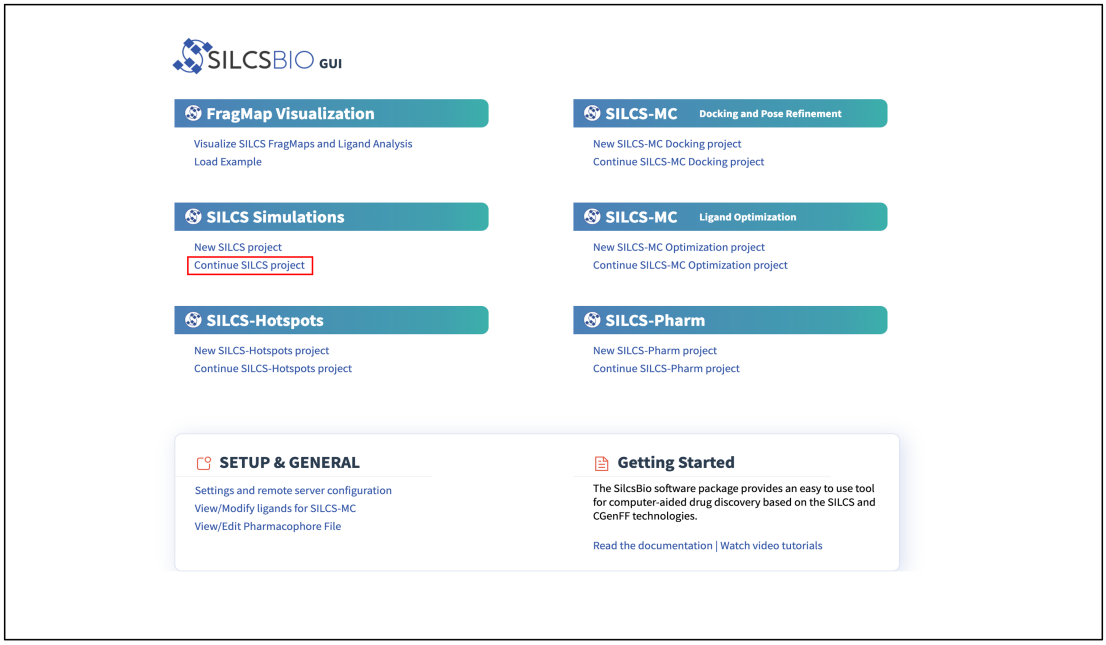
Processing the GCMC-MD trajectories will take 10-20 minutes, with the GUI providing updates on progress from the server during the process.



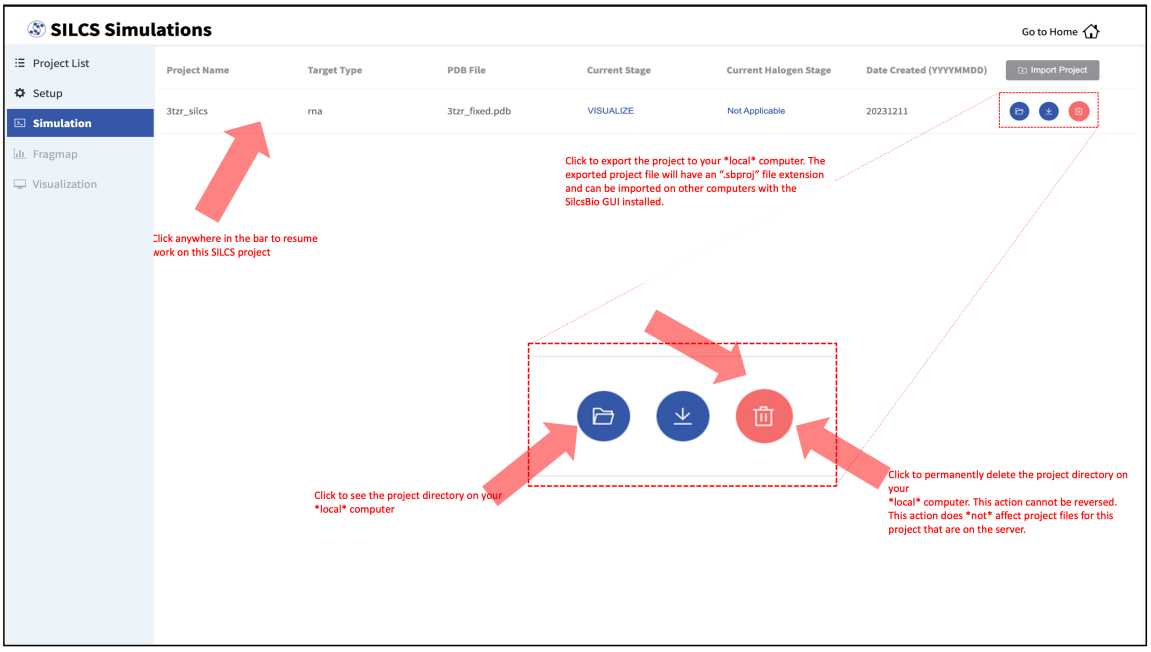
Once completed, the GUI will automatically copy the files from the server to the local computer and load them for visualization. Please see *Visualizing SILCS FragMaps* for details on how to use the SilcsBio GUI, as well as external software (MOE, PyMol, VMD), to visualize SILCS FragMaps.

7. **Continue SILCS projects:**

   To see a full listing of all of your projects, select *Continue SILCS project* from the Home page. This will show the complete list of all SILCS projects you have set up on the local machine where you are currently running the SilcsBio GUI, as well as the status of each project.



To resume work on a project or check the status of associated compute jobs you previously started, simply click the project name in the list.



Data from SILCS simulations are used to pre-compute SILCS FragMaps, which are the basis for a host of functionality. The FragMaps can be used for detecting hotspots and fragment-based drug design (*SILCS-Hotspots*), creating pharmacophore models (*SILCS-Pharm*), docking ligands and refining existing docked poses (*SILCS-MC: Docking and Pose Refinement*), and optimizing a parent ligand (*SILCS-MC: Ligand Optimization*).

For additional details on SILCS, please see *SILCS Simulations*.

Modify Ligand for SILCS-MC

Given an input parent ligand structure, the SilcsBio GUI provides an intuitive way to modify ligands and save the modified ligand structures in Mol2 format. The resulting structures can be used as input for SILCS-MC. Ligand modifications can be built independently of any other task by choosing *Modify ligand for SILCS-MC* from the Home page. For instructions on how to modify ligands, please refer to *Ligand Modifications Using the SilcsBio GUI*.

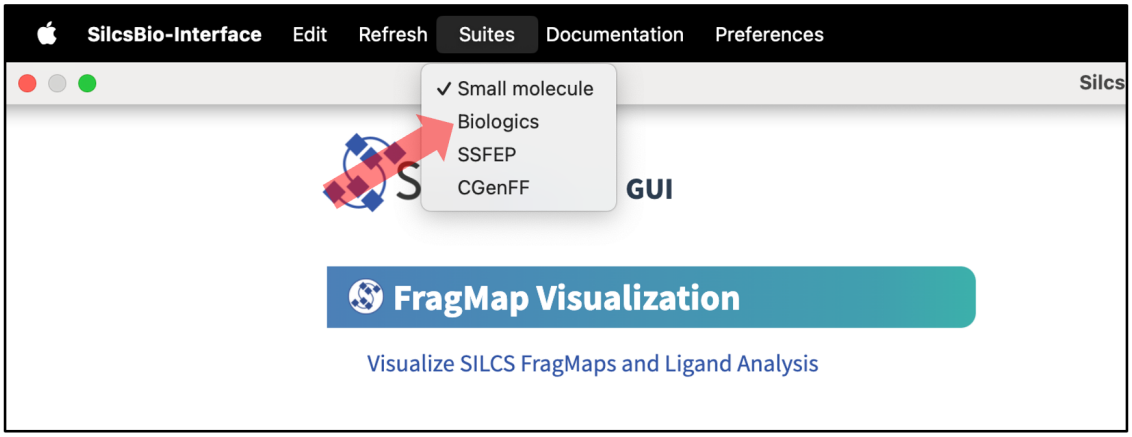
View and Edit Pharmacophore Files

Given an existing pharmacophore file, the SilcsBio GUI provides a convenient platform to view and edit pharmacophore models. The pharmacophore file viewing and editing platform can be accessed by choosing *View/Edit Pharmacophore File* from the Home page. Using this tool, users can view pharmacophore models in the context of the target protein and the corresponding FragMaps, measure distances between pharmacophore features, select or deselect pharamacophore features in the pharmacophore model, adjust the feature radii, and save the modified pharmacophore model into a new pharmacophore file. For instructions on how to view and edit existing pharmacophore files, please refer to *View and Edit Pharmacophore Files*.

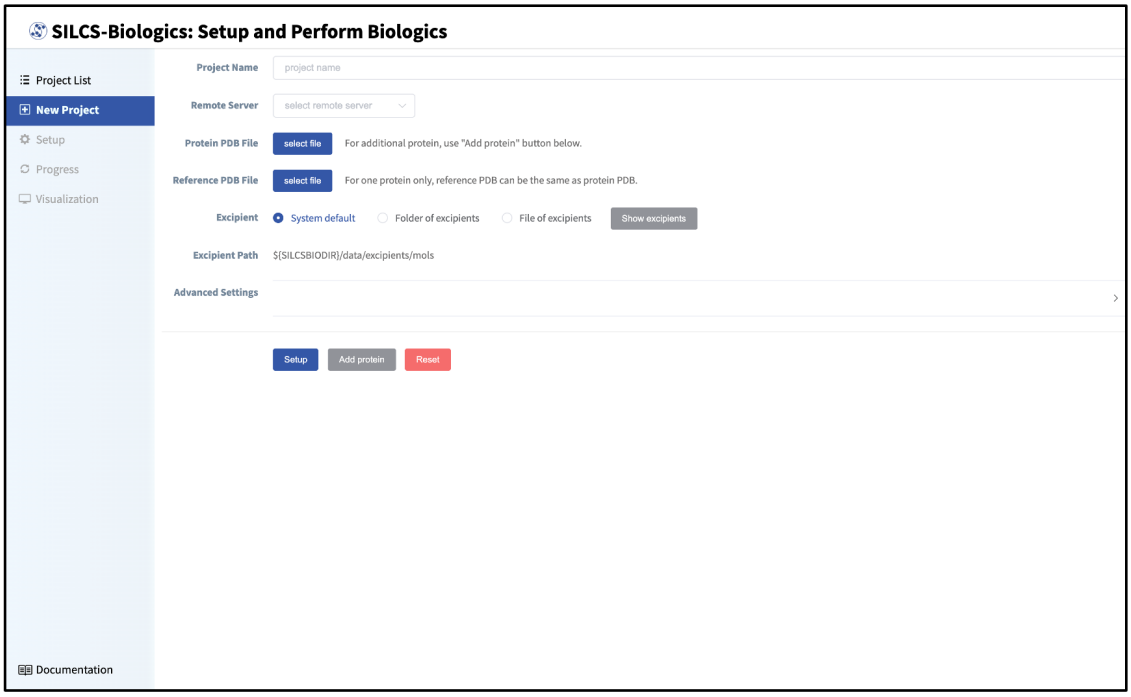
5.1.5 Biologics Suite Quickstart

SILCS-Biologics builds on the SILCS technology to facilitate the screening and selection of excipients for biologics formulations. SILCS-Biologics combines the distribution of the probability of residues participating in protein–protein interactions (PPI) on the entire protein surface, determined through SILCS-PPI, with the distribution of excipients, buffers, and monoions on the surface of the protein, determined through SILCS-Hotspots. This combination allows for excipients that may block PPI, which may contribte to protein aggregation or increased viscosity, to be identified and analyzed. Such information can be used to facilitate the selection of excipients, especially in formulations that require high protein concentrations. For more information on SILCS-Biologics, please refer to *SILCS-Biologics Background*.

SilcsBio GUI users can access the SILCS-Biologics Suite by selecting *Suites* → *Biologics* from the menu bar.



After accessing the SILCS-Biologics Suite, the user will directly enter the *New Project* window. Other windows can be selected by hovering over the left sidebar and clicking on the desired option.



From the sidebar menu, SilcsBio GUI users can

- Begin a new SILCS-Biologics project (*New Project*)
- Continue or monitor existing SILCS-Biologics projects (*Project List*)

If continuing a SILCS-Biologics project (*Project List*), SilcsBio GUI users can also

- Finalize the setup for an existing SILCS-Biologics project (*Setup*)
- Monitor the progress of an existing SILCS-Biologics project (*Progress*)
- Visualize the results of an existing SILCS-Biologics project (*Visualization*)

Details on how to use SILCS-Biologics through the SilcsBio GUI are provided below.

SILCS-Biologics Using the GUI

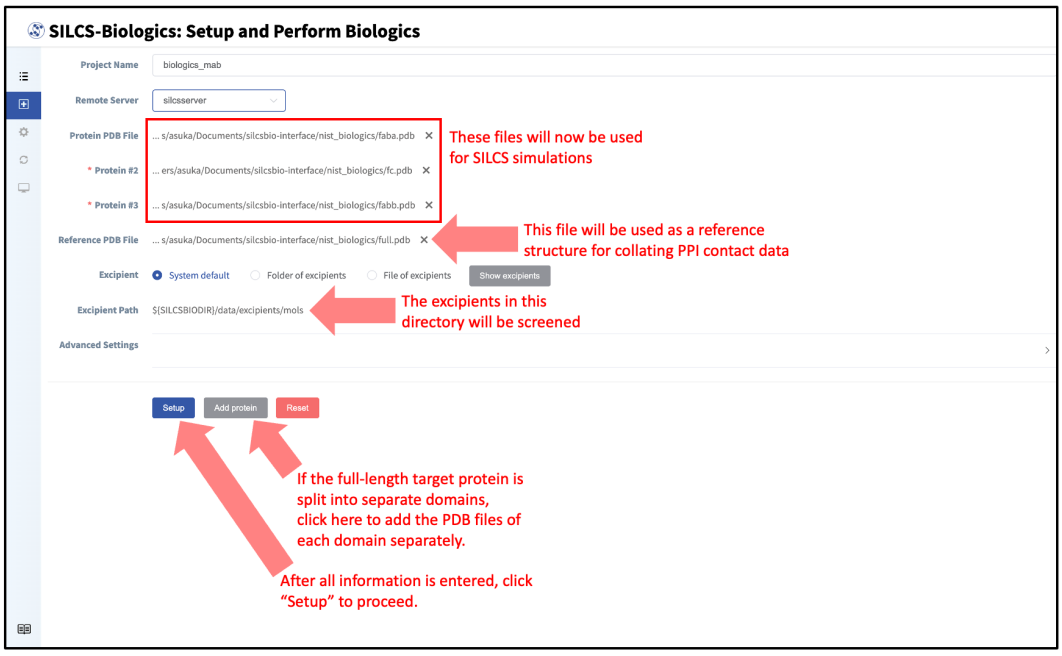
To begin a new SILCS-Biologics project, follow these steps:

1. **Begin a new SILCS-Biologics project:**

   By default, the user will directly enter the *New Project* page in which new SILCS-Biologics projects can be initiated. If the user is in a different page, the *New Project* page can be accessed by hovering over the left sidebar and selecting *New Project*.

2. **Enter a project name, select the remote server, and upload input files:**

   Enter a project name and select the remote server where compute jobs will run. Next, select a PDB file containing the coordinates of the target protein as described in *File and Directory Selection*.



   If the target protein is large and contains distinct domains (e.g., monoclonal antibodies), the full-length protein may be split into its separate domains for computational expediency. In this case, each domain should be entered as a separate PDB file. To add additional PDB files, click the "Add protein" button at the bottom of the page for each additional PDB file.

Additionally, enter a reference structure of the full-length target protein. This structure will be used a reference structure for collating PPI contact data as well as for excluding surface-exposed amino acids in individual input protein domains that are in fact buried in the context of the full-length protein. If only one PDB file was entered for "Protein PDB File", then the same PDB file should be used for "Reference PDB File".

Note: If you do not have the full-length structure of the target protein, but only have the structures of individual domains, then the full-length protein structure should be constructed using other molecular modeling tools (such as homology modeling software). The modeled full-length structure can then be used as the "Reference PDB File".

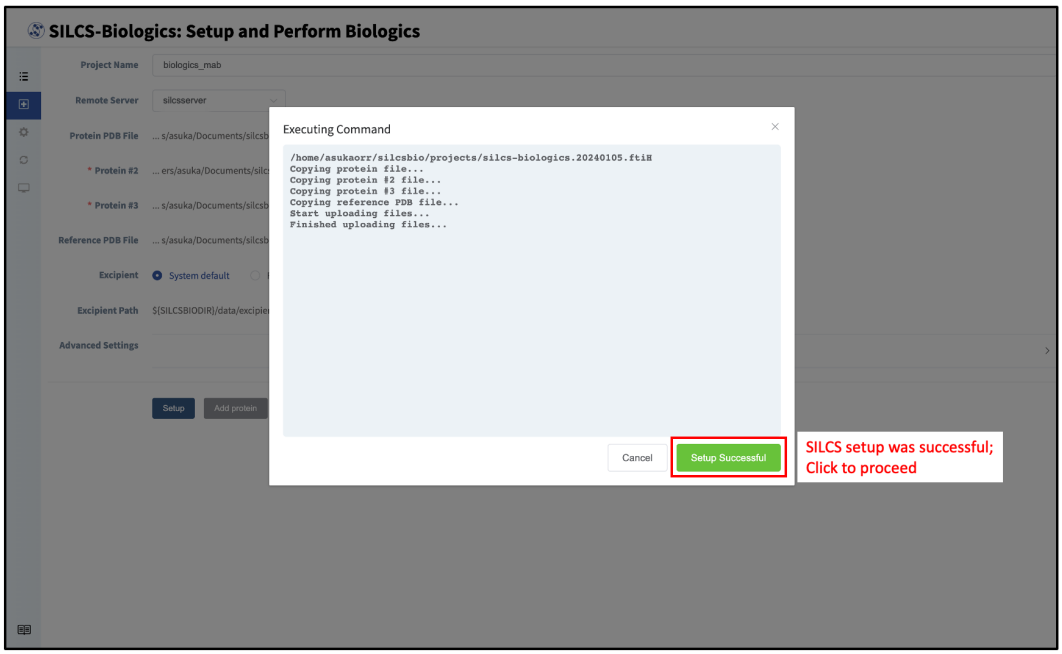
Alternatively, the positions of the individual domains in the context of the full-length protein can be estimated through a simple structural alignment to a known full-length homologous crystal structure, and the resulting superimposed structure of the individual domains can be saved and used as the "Reference PDB File".

Warning: Dividing the target protein into domains entails independent SILCS simulations for each of the domains. E.g., for SILCS-Biologics runs with `prot1`, `prot2`, and `prot3` specified (3 domains) 3 $\times$ 10 SILCS simulations will be performed rather than 1 $\times$ 10 SILCS simulations. If sufficient compute nodes are available to run all SILCS simulations for all domains simultaneously, the SILCS simulations for the target protein will be completed sooner using the domain approach than running SILCS simulations of the full-length protein. However, using the domain approach may consume more CPU/GPU hours.

The SilcsBio GUI provides a default set of excipients for screening ("System default"). If a custom set of excipients is desired, click the "Folder of excipients" (if excipients are stored as individual Mol2 or SD files in a folder) or "File of excipients" (if all excipients are stored in one SD file) radio button next to *Excipient*.

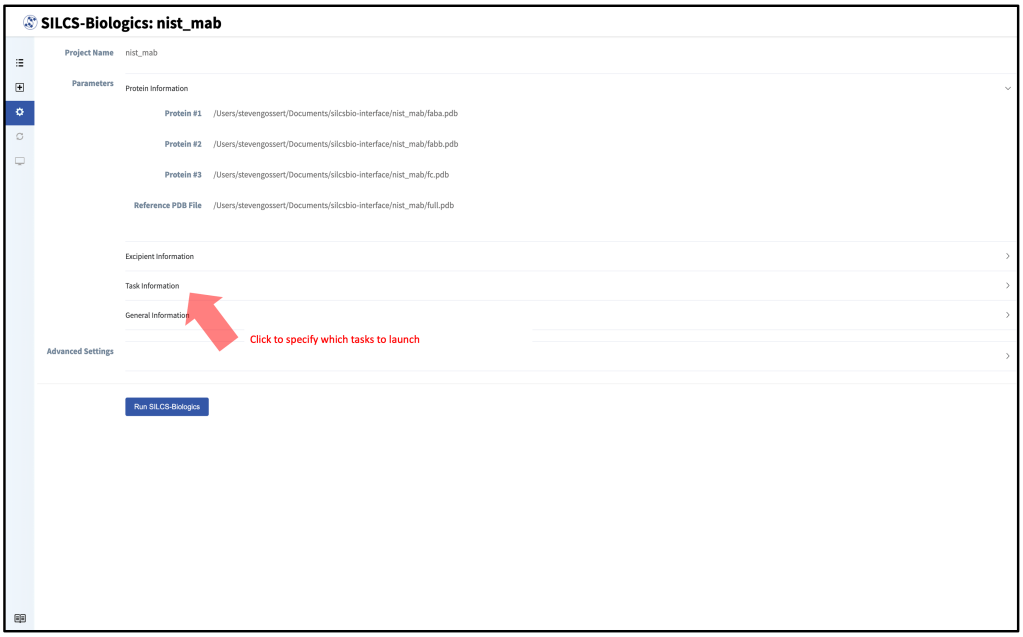
3. **Set up SILCS-Biologics simulations:**

   Once all information is entered, click the "Setup" button at the bottom of the page. The GUI will contact the remote server and upload the input files specified in the previous step. A green "Setup Successful" button will appear once the process has successfully completed. Press this button to go to the next step.



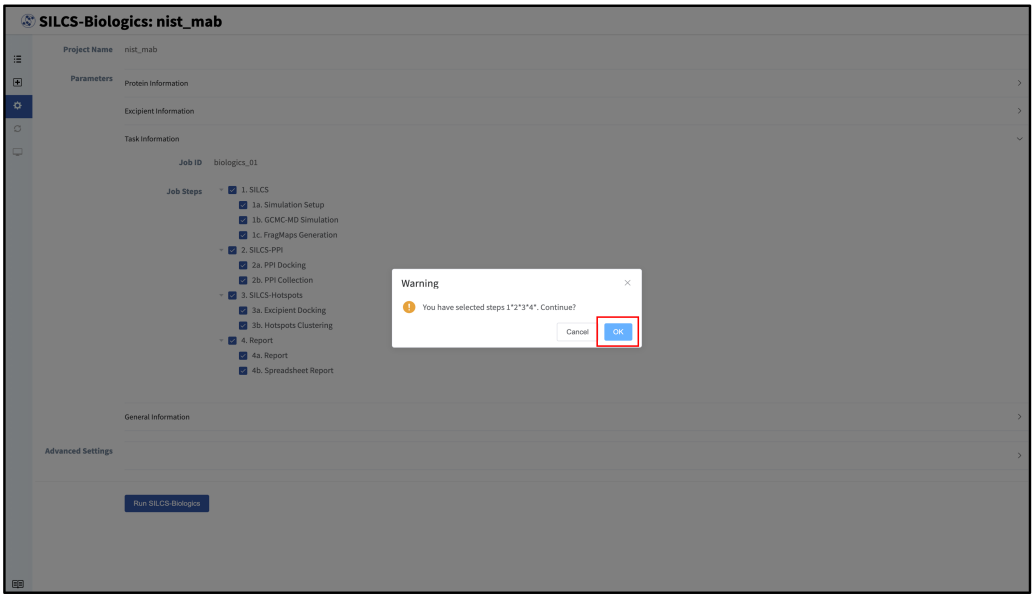
4. **Launch SILCS-Biologics simulations:**

   Your SILCS-Biologics simulations can now be started by clicking the "Run SILCS-Biologics" button at the bottom of the page. SILCS-Biologics may be run in a stepwise fashion if desired. To view and toggle on/off SILCS-Biologics tasks, click "Task Information".

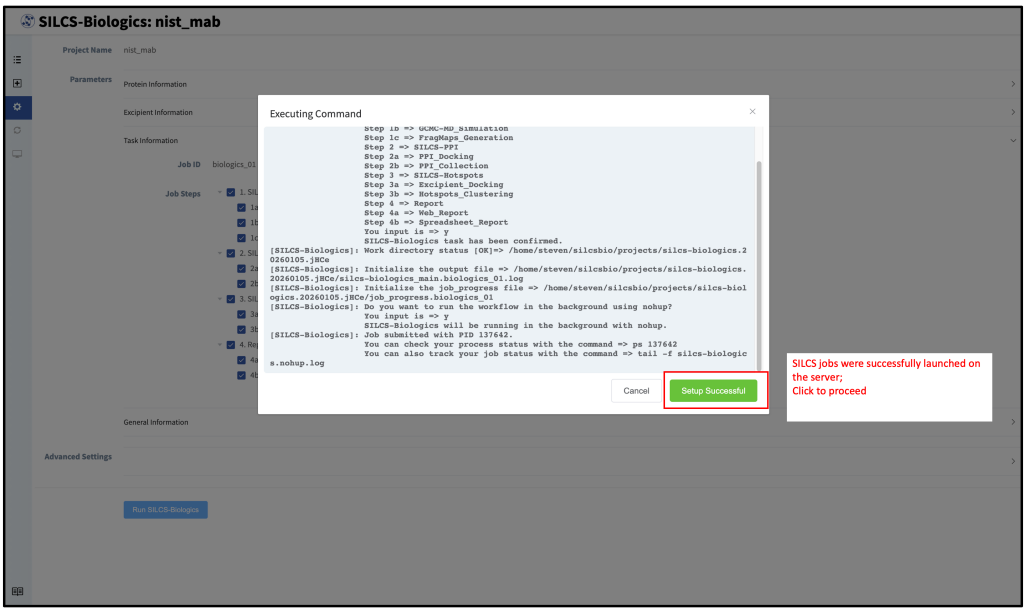


   Finer control at the level of the smaller subtasks can also be requested. Note that the tasks and subtasks must be run sequentially (e.g., Step 1a must be run before Step 1b, and Step 1* must be run before Step 2*). By default, all tasks will be run.

After clicking the "Run SILCS-Biologics" button, the SilcsBio GUI will prompt the user to confirm the tasks to be submitted with a warning window. Click "OK" to confirm the selected tasks. To return to the task selection page, click "Cancel".



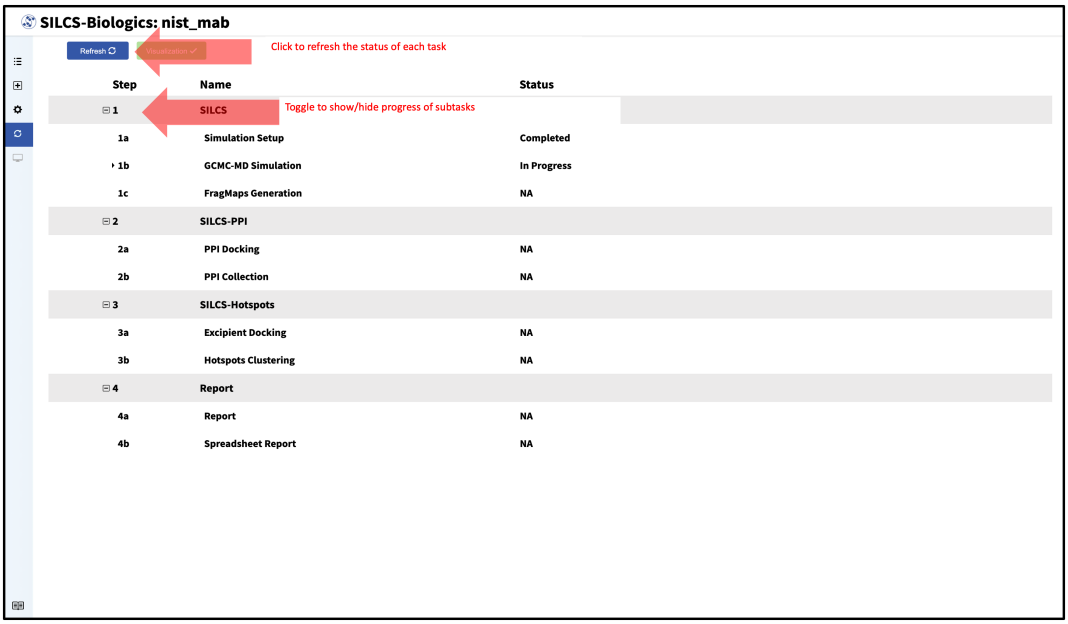
After confirming the tasks to be performed with clicking "OK" on the warning window, compute jobs will be submitted to the queueing system on the server.



A green “Setup Successful” button will appear once the jobs are successfully launched on the server. Click this button to proceed.

5. **Monitor SILCS-Biologics tasks:**

   The progress of each SILCS-Biologics task can be monitored by clicking “Progress” in the sidebar menu. Alternatively, the “Progress” page can be accessed by going to the “Project List” page through the sidebar menu and clicking on the desired SILCS-Biologics project if the project’s SILCS-Biologics simulations have already been submitted. The SilcsBio GUI will also automatically take you to the “Progress” page after clicking “Setup Successful” in the previous step.



In the "Progress" page, the status of each task can be refreshed by clicking the "Refresh" button at the top left of the page. The progress of each subtask can also be viewed by toggling "+/–".

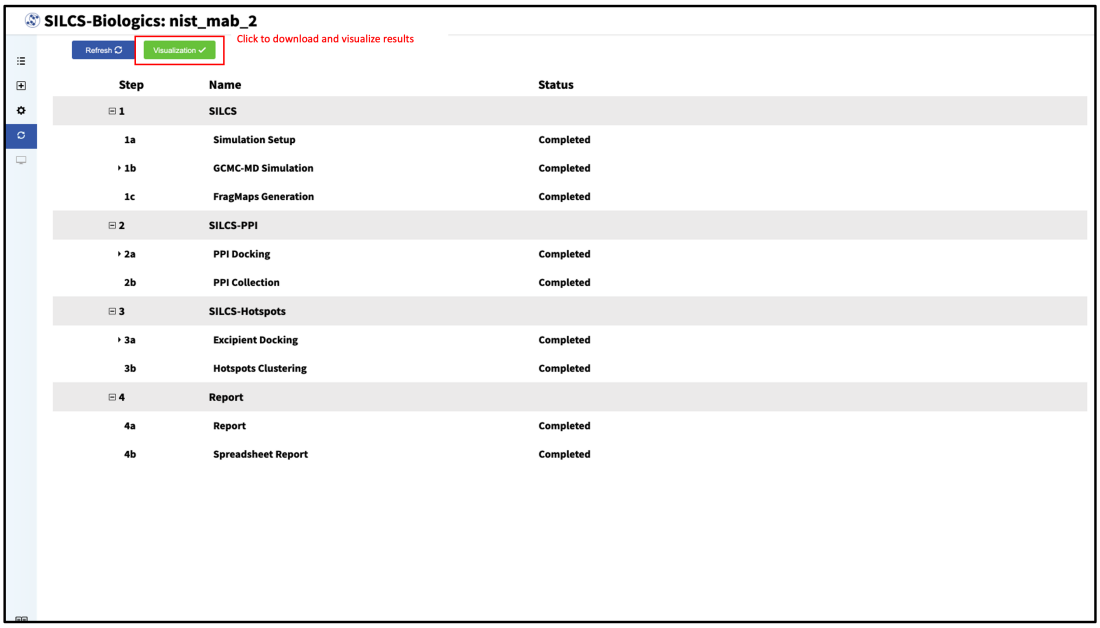
For each subtask step involving simulations (SILCS simulations, SILCS-PPI docking, and SILCS-Hotspots excipient docking), the progress of individual simulations can also be viewed by clicking the dropdown menus and toggling "+/–".

6. **Download and visualize SILCS-Biologics results:**

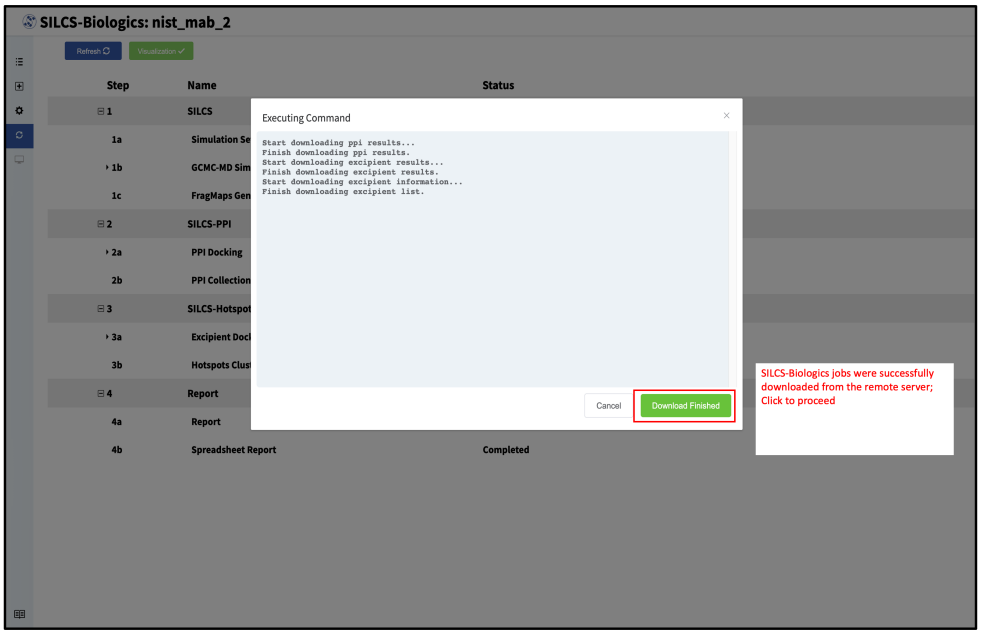
   Once all SILCS-Biologics tasks are complete, the "Status" column on the Progress page for

all tasks will be marked “Completed” and the green “Visualization” button at the top of the page will become active.

Click the green “Visualization” button to download the results from the remote server.

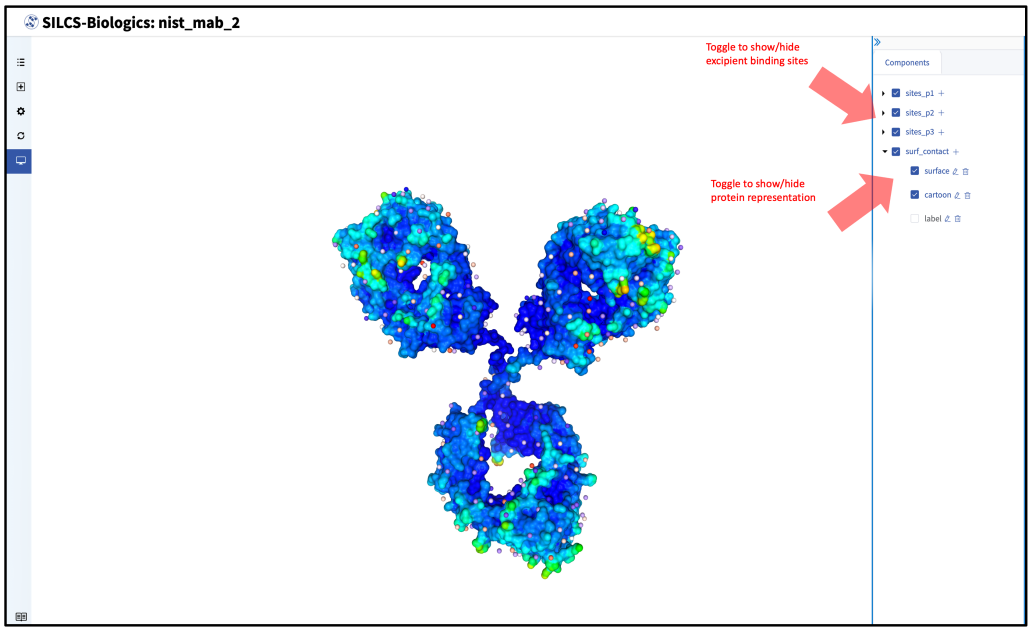


A window will appear with the download progress. After the results are successfully downloaded, the GUI will automatically update with a green “Download Finished” button. Click the “Download Finished” button to visualize the results.



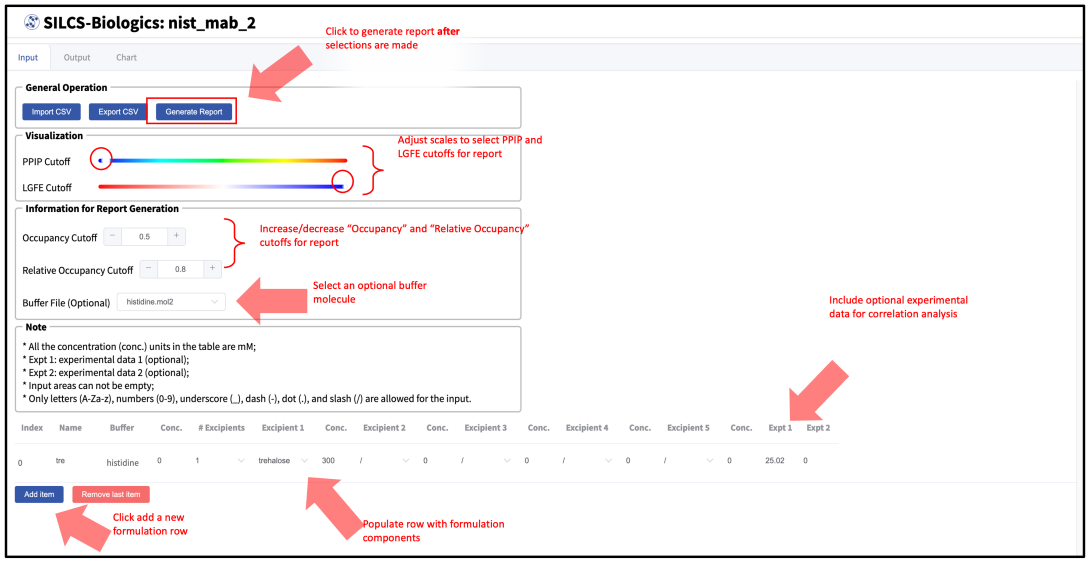
The SILCS-Biologics results (protein–protein interaction preference [PPIP] and excipient binding sites) can now be visualized for the target protein. The target protein is shown in sur-

face representation, by default, with regions of high PPIP (regions prone to self-aggregation) colored red (PPIP score of 1) and regions with low PPIP colored blue (PPIP score of 0). In the example case shown in the figures, SILCS-Biologics was applied to NISTmAb, which is not prone to aggregation. This is reflected by the primarily blue surface shown. Excipient binding sites are shown in vdW representation, with more energetically favorable binding sites colored red and less energetically favorable (site LGFE of greather than -2 kcal/mol) colored blue. Representations of the protein can be customized under "surf_contact". Representations of the excipient binding sites can be customized under "sites_p<protein number>", with <protein number> indicating the protein PDB file entered. In the example case shown in the figures, sites_p1 corresponds to faba.pdb ("Protein PDB File"), sites_p2 corresponds to fc.pdb ("Protein #2"), and sites_p3 corresponds to fabb.pdb ("Protein #3").



7. **Generate formulation report:**

   The SILCS-Biologics data can help connect experimental observables to the molecular details of protein–excipient interaction. In doing so, these data can help both rationalize known trends for existing data on excipients and guide the selection of new excipients for additional testing. With regard to the former, the most straightforward approach is to compare available experimental data to SILCS-Biologics metrics and note correlations. The SilcsBio GUI provides a platform to generate formulation reports and chart correlations between experimental data and SILCS-Biologics metrics.



To generate a report, first enter the details on formulations of interest at the bottom of the page. These details are:

- (1st column, "Name") the name of the formulation,
- (2nd column, "Buffer") the buffer molecule if applicable,
- (3rd column, "Conc.") the concentration in mM of the buffer molecule if applicable,
- (4th column, "# Excipients") the number of excipients in the formulation excluding the buffer molecule,
- (5th column, "Excipient 1") the first excipient in the formulation,
- (6th column, "Conc.") the concentration in mM of the first excipient in the formulation,
- (7th, 9th, 11th, or 13th column, "Excipient N") the Nth excipient in the formulation if applicable,
- (8th, 10th, 12th, or 14th column, "Conc.") the concentration in mM of the Nth excipient if applicable,
- (15th column, "Expt 1") the first experimental data for the formulation if available, and
- (16th column, "Expt 2") the second experimental data for the formulation if available.

To add additional formulations to the analysis, click the "Add item" button at the bottom of the page. To remove the last formulation entered, click the red "Remove last item" button. Alternatively, this information can be uploaded all at once by importing a a CSV file using the "Import CSV" button under "General Operation" at the top of the page. The formulations can additionally be saved into a CSV file using the "Export CSV" button in the same location.

Next, specify cutoff values to include/exclude bindings sites for analysis in the formulation report.

The PPIP and LGFE cutoffs can be selected using the selection scale under "Visualization". The colors on the scale correspond to the colors shown in the visualization window. Using the LGFE criterion, for an excipient binding site to be counted, the excipient(s) in a given formulation must have an LGFE less than the LGFE cutoff. Using the PPIP criterion, for an excipient binding site to be counted, the residues in proximity to the binding site must have a PPIP greater than the PPIP cutoff.

The occupancy and relative occupancy cutoffs can be selected under "Information for Report Generation". Using the occupancy criterion, for an excipient binding site to be counted, the excipient(s) in a given formulation must have an occupancy greater than the occupancy cutoff. Using the relative occupancy criterion, for an excipient binding site to be counted, the excipient(s) in a given formulation must have a relative occupancy greater than the relative occupancy cutoff.

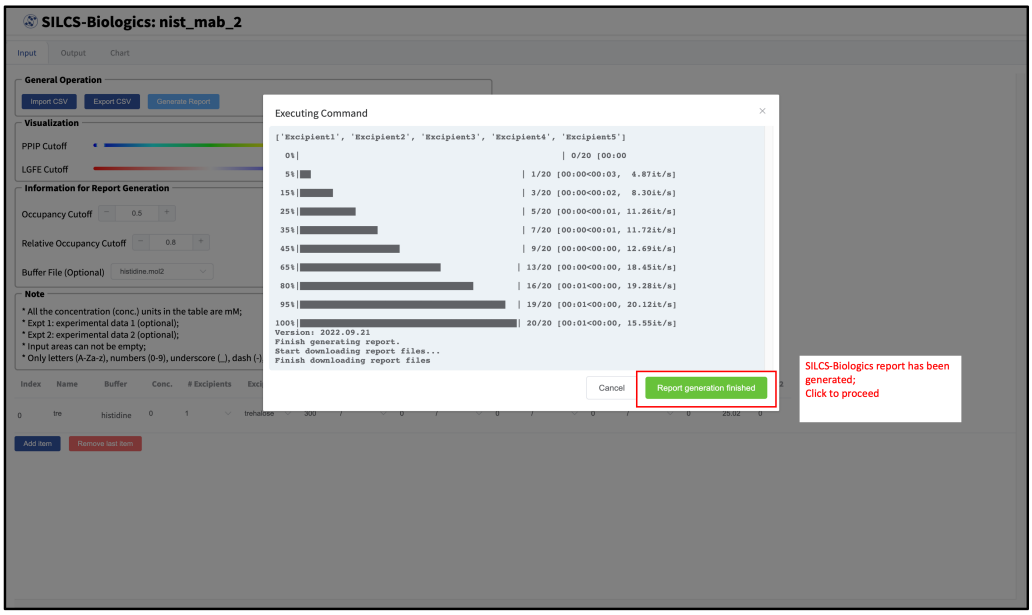
Note: The occupancy of an excipient at a given binding site is a function of its LGFE and concentration:

$$\text{occupancy} = [L]/([L] + K_d)$$

where K_d is the dissociation constant such that $LGFE = RT \ln K_d$ and $[L]$ is the concentration of the excipient in the formulation with unit mM. The occupancy has a range of 0 to 1 with 1 indicating that the excipient has a high probability of binding at the site and 0 indicating that the excipient has minimal probability of binding at the site.

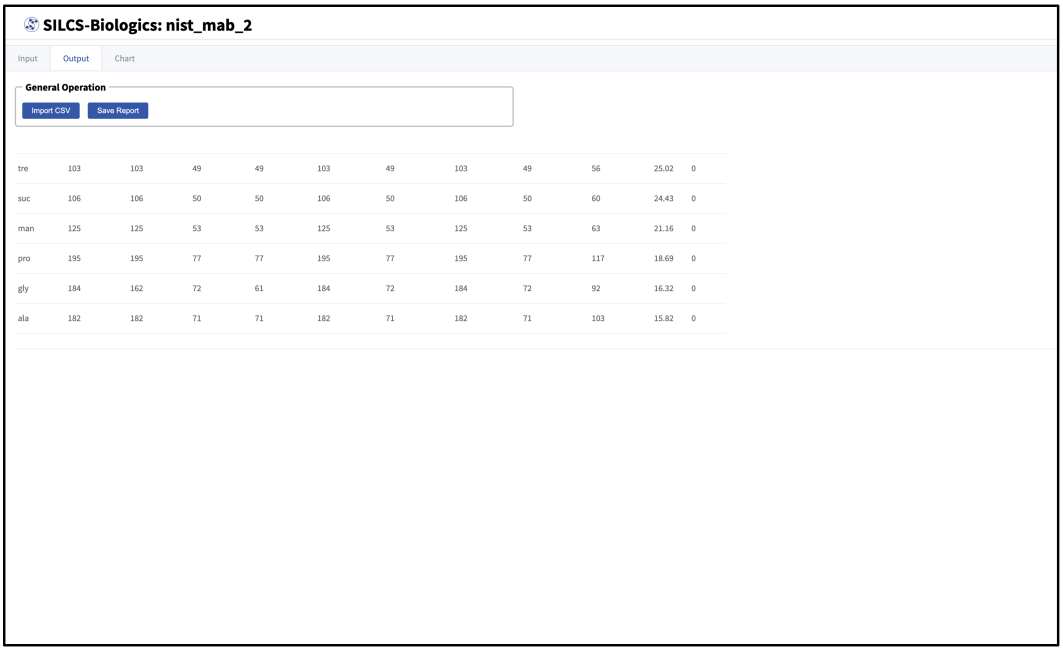
Finally, select the Mol2 file of the buffer molecule if applicable in the dropdown menu under "Information for Report Generation".

Once all selections have been made, click the "Generate Report" button at the top of the page, under "General Operation". The SilcsBio GUI will initiate calculations to generate SILCS-Biologics metrics based on the specified formulations and cutoffs. Upon completion of the calculations and report generation, a green "Report generation finished" button will appear. Click on the green "Report generation finished" button to view the output.



8. **View formulation report output:**

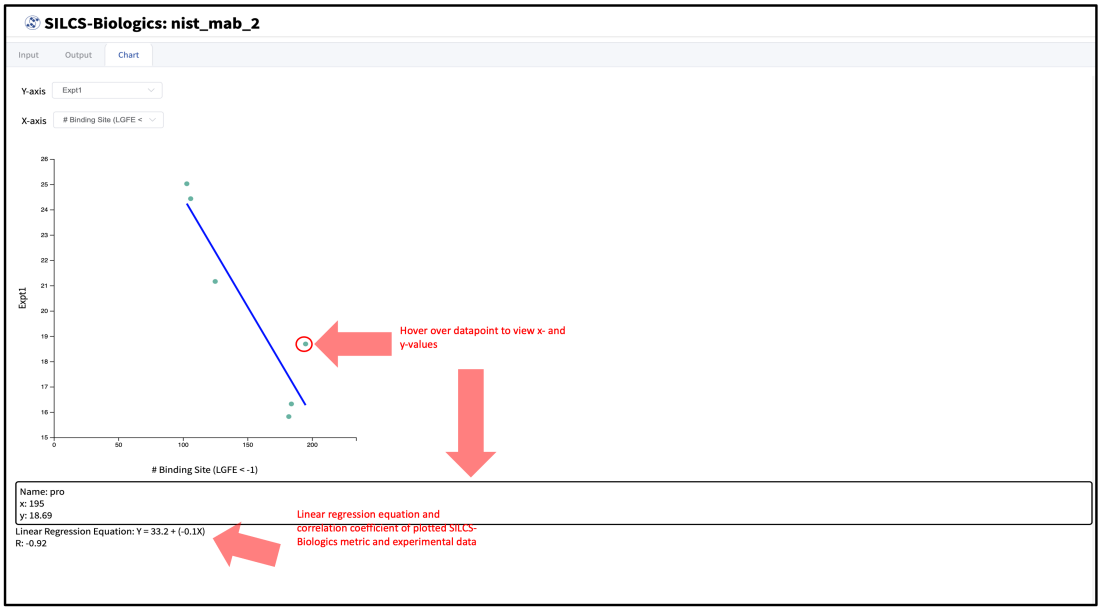
   After the calculations are completed and the report is generated, the GUI will automatically display a table of SILCS-Biologics metrics for each formulation entered in the previous step. Data from an existing CSV output can also be uploaded using the "Import CSV" button in the "Output" tab.



   To plot the SILCS-Biologics metrics against experimental data, click the "Chart" tab and select the Y-axis (choose between experimental data entered for "Expt1" and "Expt2" in the previous step) and X-axis (choose SILCS-Biologics metrics generated based on specifications

selected in the previous step).

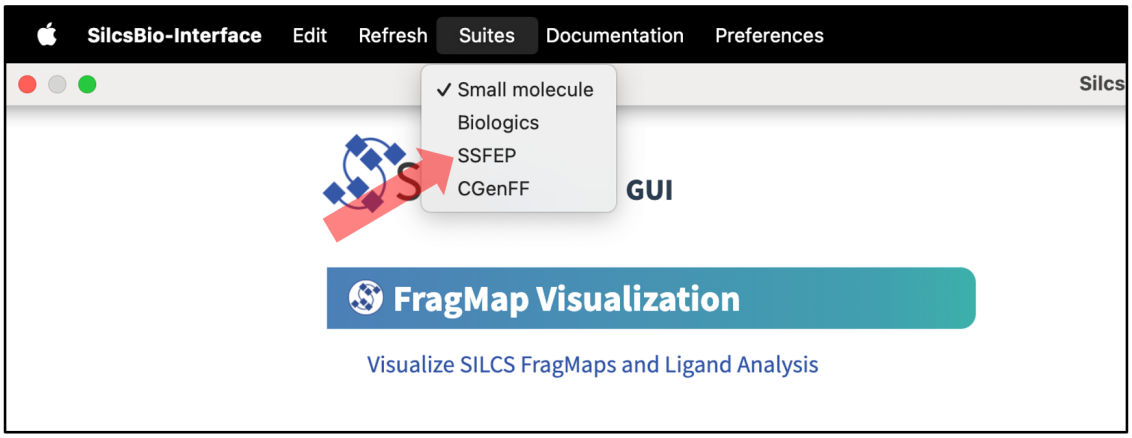
The plot will appear automatically with the linear regression equation and correlation between the experimental data and the SILCS-Biologics metric ("R") below the plot. Specific values and the formulation name of each datapoint can be viewed by hovering over the datapoint.



5.1.6 SSFEP Suite Quickstart

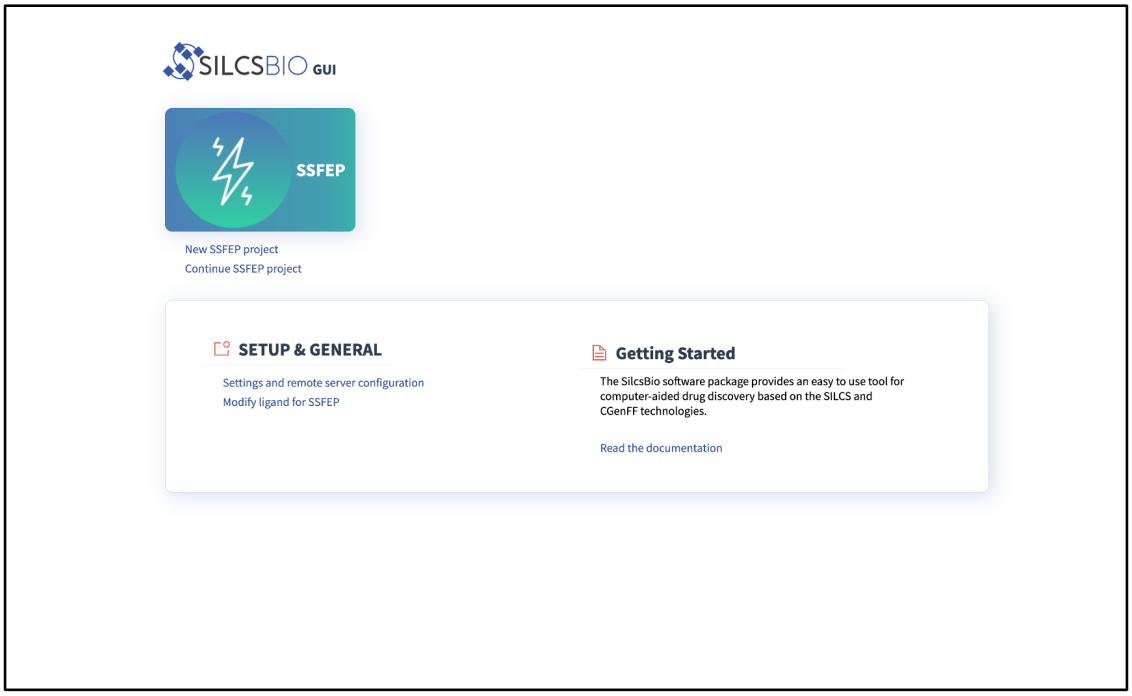
Single Step Free Energy Perturbation (SSFEP) is a rapid method for evaluating relative binding affinities, enabling the assessment of numerous modifications to a parent ligand. Additional information on SSFEP can be found in *SSFEP: Single Step Free Energy Perturbation*.

SilcsBio GUI users can access the SSFEP Suite by selecting *Suites* → *SSFEP* from the menu bar.



The SSFEP Suite Home page provides easy access for users to

- Begin a new SSFEP project (*New SSFEP project*)
- Continue an existing SSFEP project (*Continue SSFEP project*)



The SSFEP Suite also provides convenient access to general tools under *SETUP & GENERAL* at the bottom of the page.

- *Build ligand modifications for SSFEP*
- *Settings and remote server configuration*

For additional information on these tools, please click on their links. Details on how to use the SSFEP Suite in the SilcsBio GUI are provided below.

SSFEP Simulations Using the GUI

To begin a new SSFEP project, follow these steps:

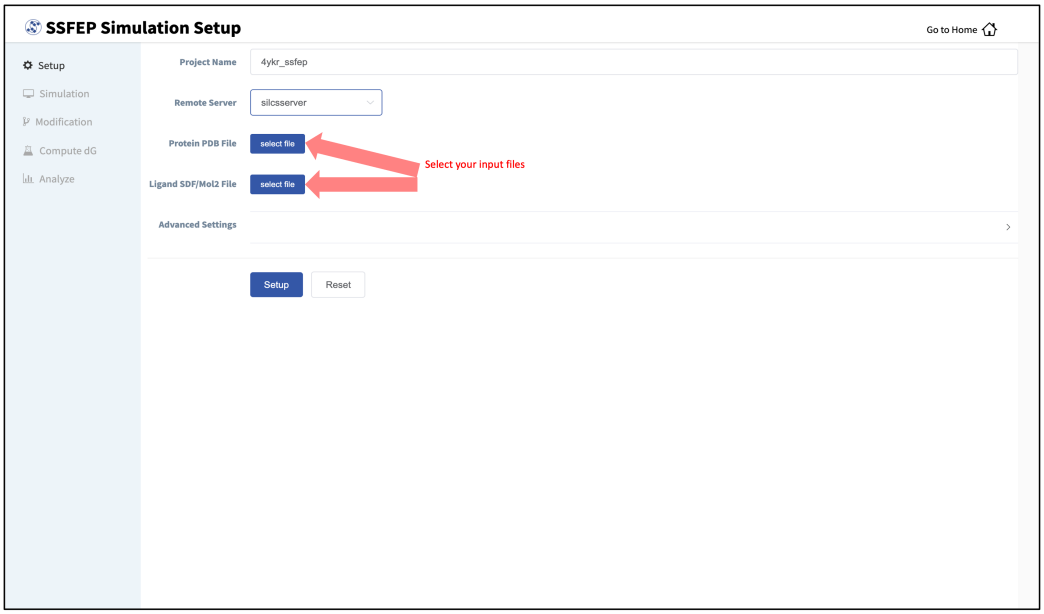
1. **Begin a new SSFEP project:**

   Select *New SSFEP project* from the Home page.

2. **Enter a project name, select the remote server, and upload input files:**

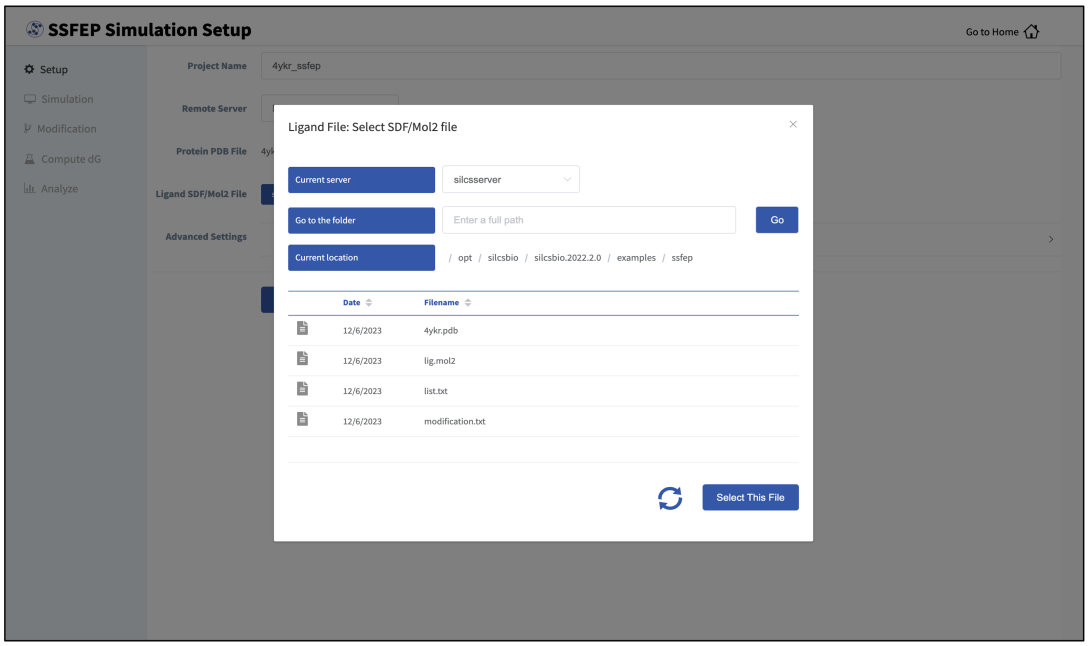
   Enter a project name, and select the remote server where compute jobs will run.

Note: Typical SSFEP simulations produce output files in excess of 20 GB, so please select a project location file folder with appropriate storage capacity. The project location can be changed by entering the "Settings and remote server configuration" page from the "Home" page and changing the "Project Directory" field as described in *Remote Server Setup*.



Next, select a protein PDB file and a ligand file as described in *File and Directory Selection*. The ligand should be aligned to the binding pocket in the accompanying protein PDB file. We recommend cleaning your PDB file before use, incuding keeping only those protein chains that are necessary for the simulation, removing all unnecessary ligands, renaming non-standard residues, filling in missing atomic positions, and, if desired, modeling in missing loops.

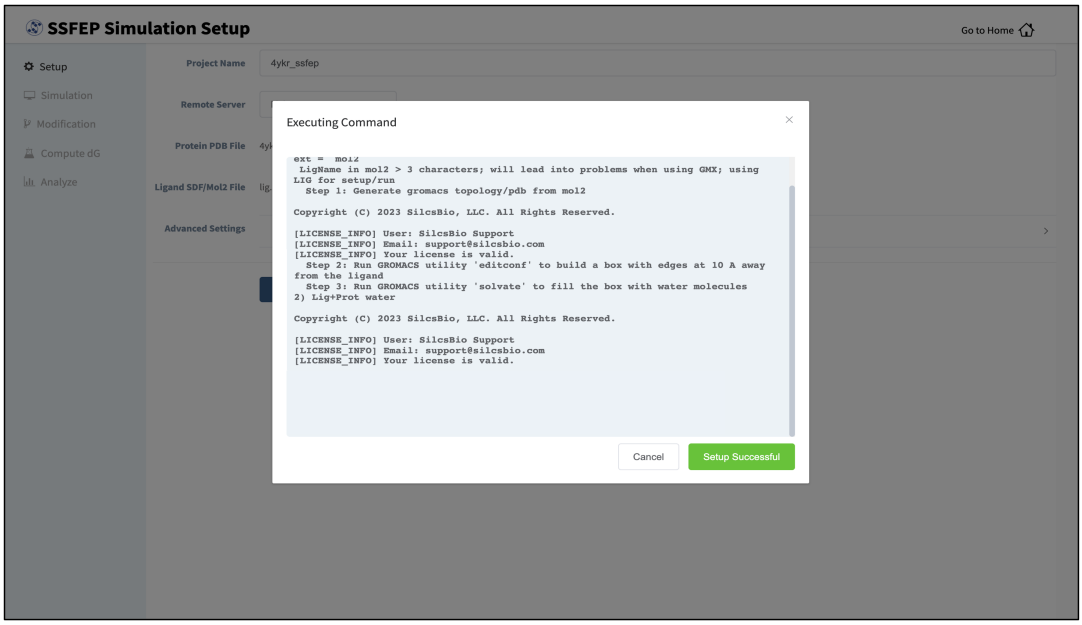
If the GUI detects missing non-hydrogen atoms, non-standard residue names, or non-contiguous residue numbering, it will inform the user and provide a button labeled "Fix?". If this button is clicked, a new PDB file with these problems fixed and with `_fixed` added to the base name will be created and used in the SSFEP simulation.



3. **Set up SSFEP simulations:**

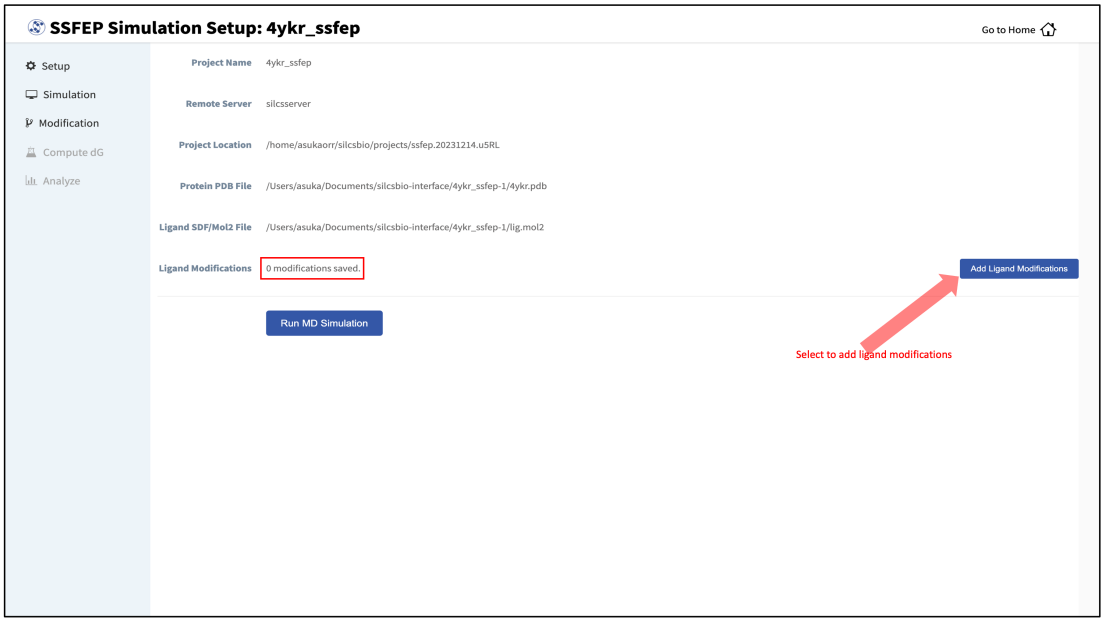
   Once all information is entered correctly, press the "Setup" button at the bottom of the page. The GUI will contact the remote server and perform the SSFEP setup process.

   During setup, the program automatically performs several steps including building the topology of the simulation system and creating metal-protein bonds if metal ions are found. To complete the entire process may take up to 10 minutes depending on the system size. A green "Setup Successful" button will appear once the process has successfully completed. Press this button to proceed.



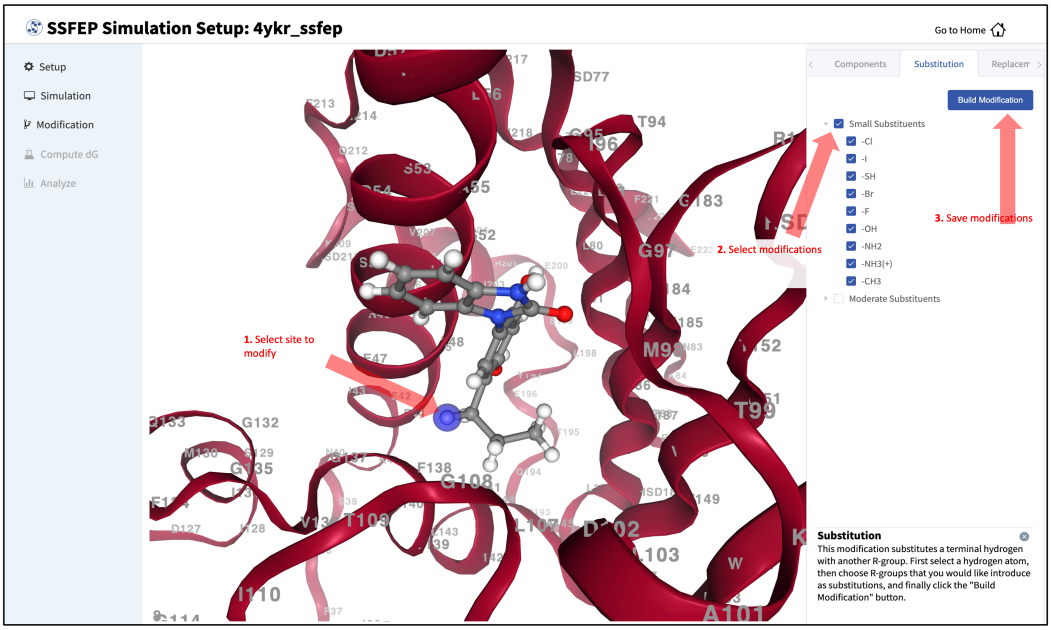
4. **Select ligand modifications:**

   You can now prepare your ligand modifications with the "Add Ligand Modifications" button.



   There are two major modification types, Substitution and Replacement, available in the GUI. Substitution is used to substitute a hydrogen with another functional group. Replacement is used to replace an atom in a ring with another functional group that preserves the ring.

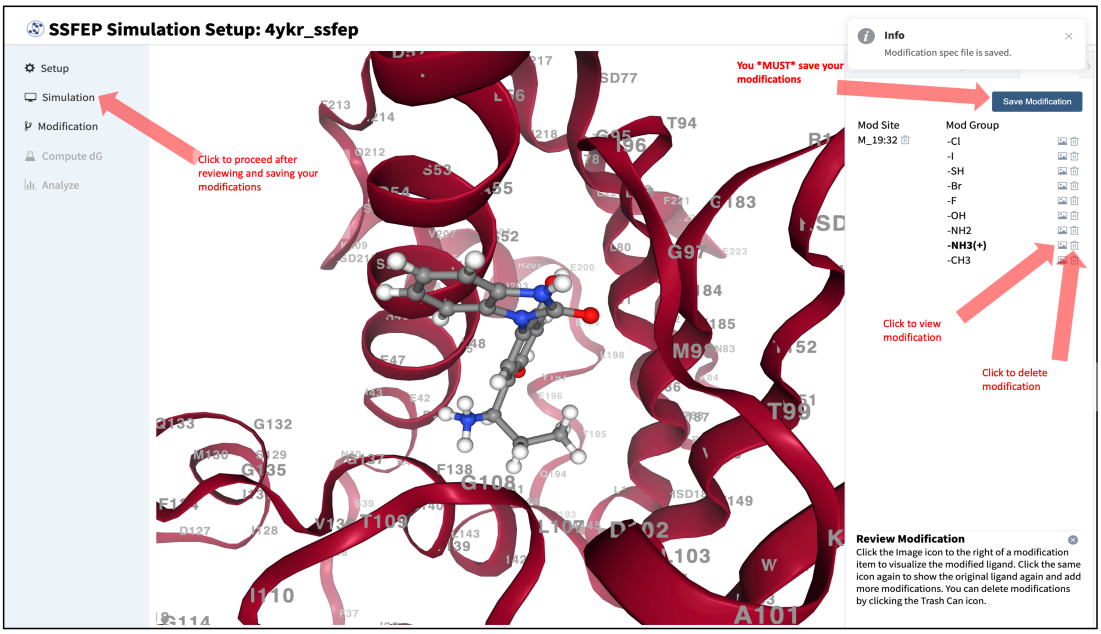
   In the visualization window, select the atom to be modified. Then, select your desired modifications from the "Substitution" or the "Replacement" tab in the right-hand panel. Pressing the "Save" button in the panel will update your list of modifications.



The list of modification types in the GUI covers a broad range of chemical functionality. Custom modifications can be made using the Command Line Interface (CLI) as detailed in *SSFEP: Single Step Free Energy Perturbation*.

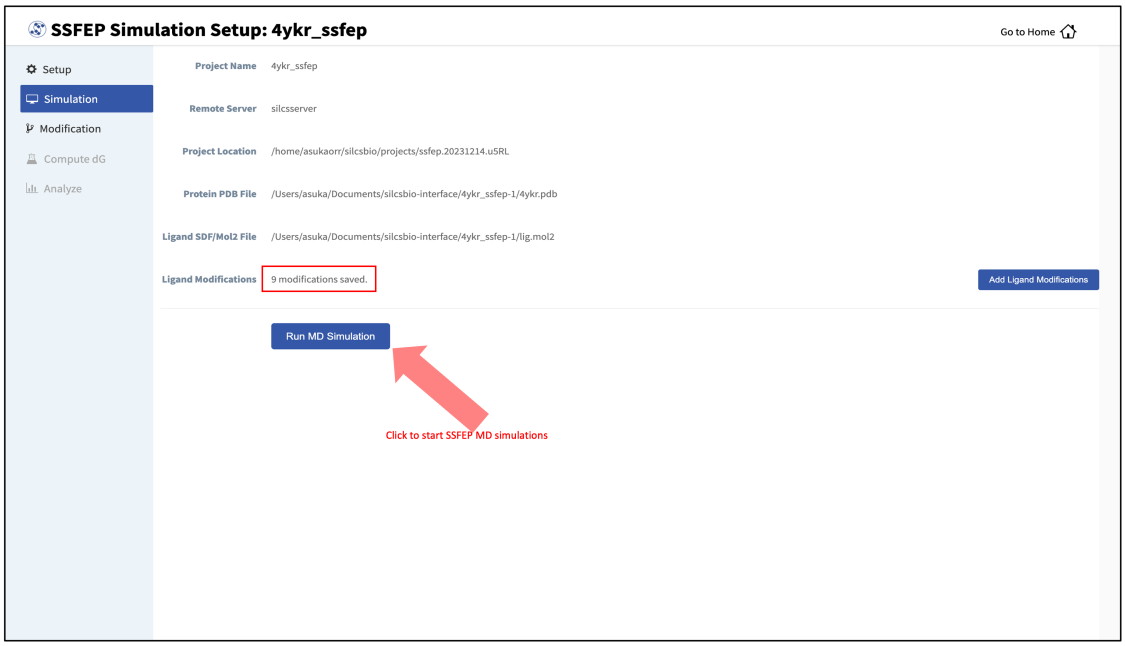
5. **Review selected ligand modifications:**

   Use the "Review" tab to confirm your desired modifications. Valid modifications will have a small Image icon as well as a small Trash Can icon. Clicking on the Image icon will show the modification in the center panel. Clicking on it again will show the parent ligand. Clicking on the Trash Can icon will delete the proposed modification from your list. You can go back to the "Substitution" and "Replacement" tabs to add to your list. Once you have completed your list of modifications, **you must press the "Save Modification" button in the "Review" tab to actually save the list of modifications for your project**.



6. **Launch SSFEP simulations:**

   Your SSFEP simulation can now be started by selecting "Simulation" from the left-hand pane and then clicking the "Run MD Simulation" button in the resulting screen. Before running, you may wish to double check that you have chosen the desired file folder on the remote server and that it has 20+ GB of storage space. There are two parts to SSFEP: a compute-intensive MD simulation and a very rapid $\Delta\Delta G$ calculation. The compute-intensive MD may take several hours and is done only once. Using the resulting MD data, the rapid $\Delta\Delta G$ calculation is able to test thousands of functional group modifications to your parent ligand in under an hour. Should you wish to test additional modifications to your parent ligand at a later time in the project, there is no need to re-run the compute-intensive MD, which makes SSFEP a very efficient method.



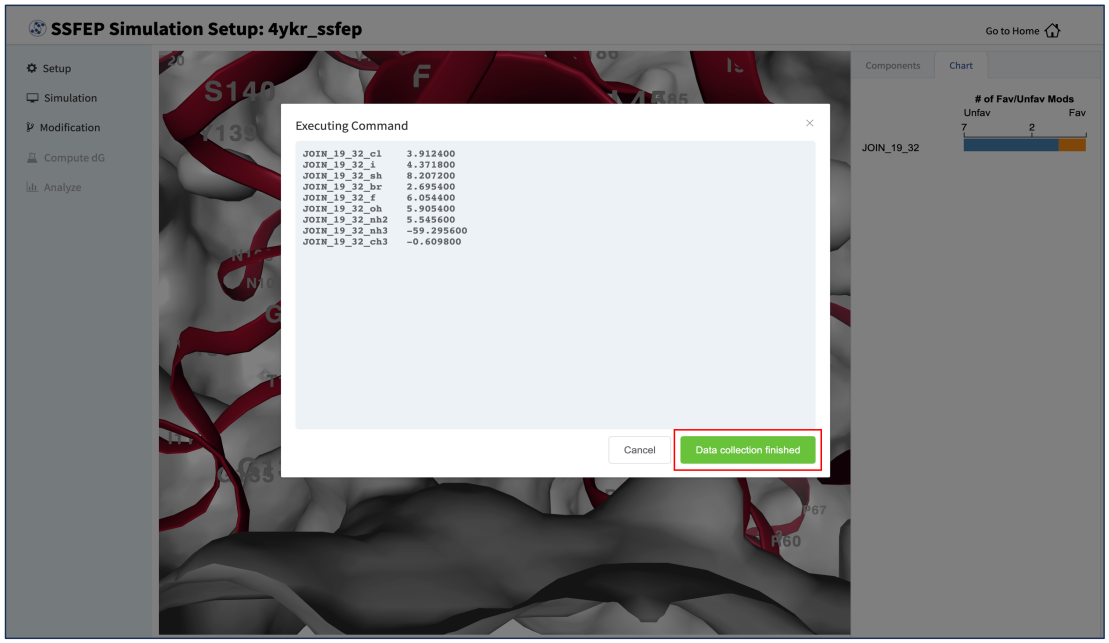
10 compute jobs will be submitted to the queueing system (five for the ligand and five for the protein–ligand complex) on your remote server. Job progress will be displayed in this same window. The status of each job is shown next to its progress bar: "Q" for queued and "R" for running, At this point, your jobs are in progress and you may safely quit the SilcsBio GUI or go back to the Home page to do other tasks.

If a job encounters an error and does not finish, a "restart" button will appear next to the status. If the restart button is used, the job will be resubmitted to the queue and continue from the last cycle of the calculation.

7. **Calculate and collect $\Delta\Delta G$ values:**

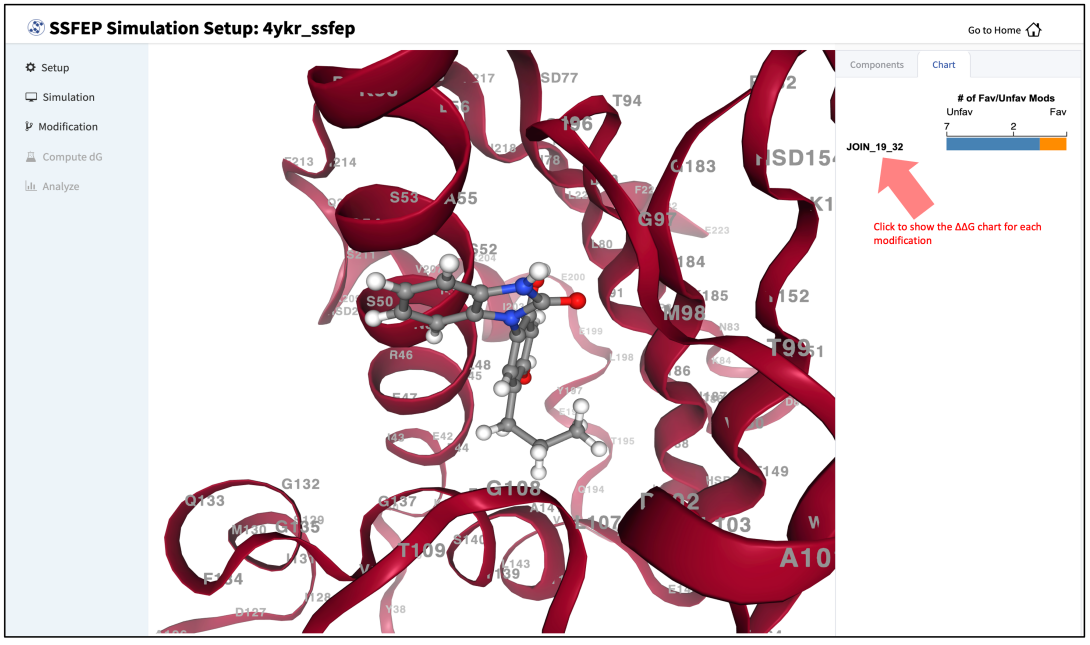
   Once the MD simulation stage has finished, calculate the $\Delta\Delta G$ values for your list of modifications by clicking on the "Compute ddG" button at the bottom of the page.

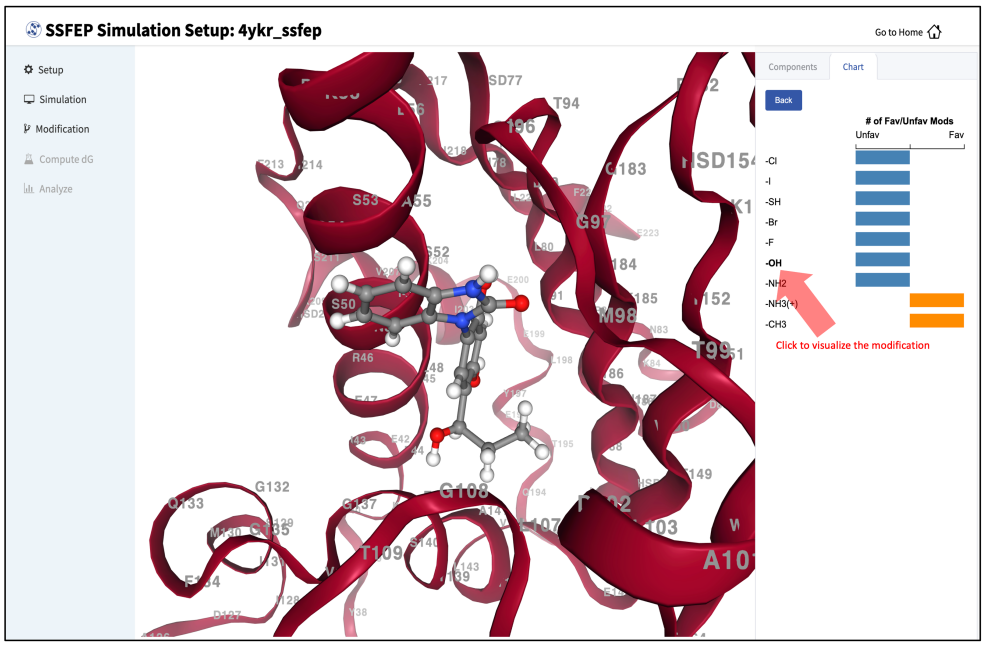
The completion of the $\Delta\Delta G$ calculations will be indicated by the appearance of the green "Show Chart" button at the bottom of the screen. Click the green "Show Chart" button to collect the data onto the local server. Once data collection is completed, a green "Data collection finished" button will appear. Clike the "Data collection finished" button to proceed.



8. **Visualize $\Delta\Delta G$ direction:**

   The predicted changes in direction of binding affinity relative to the parent ligand for each of your selected modifications are plotted to the right of the screen. You can also visualize the modifications in the protein–ligand complex.





Note: SSFEP is designed to evaluate small modifications, and results are best interpreted qualitatively. Therefore GUI-created charts indicate the predicted change in direction of the binding affinity relative to the parent ligand.

For additional details on SSFEP, please see *SSFEP: Single Step Free Energy Perturbation*.

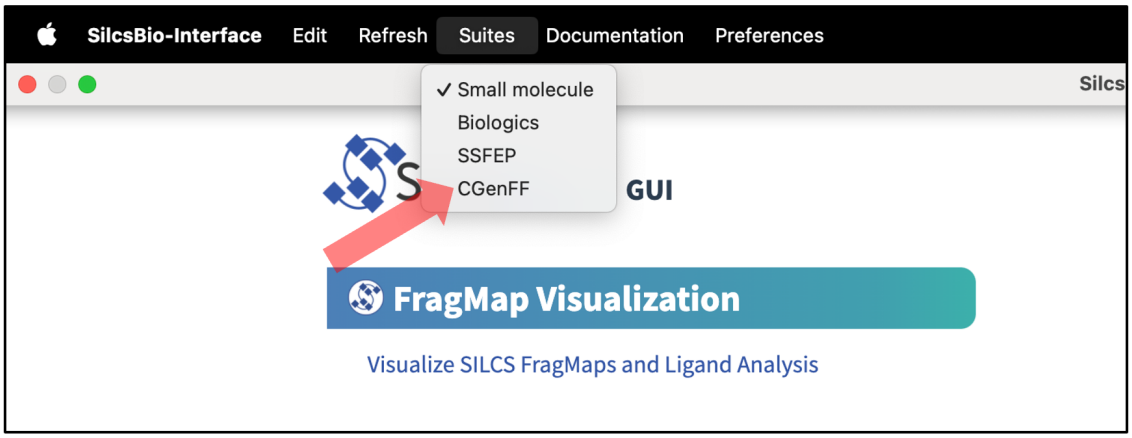
Modify Ligand for SSFEP

Given an input parent ligand structure, the SilcsBio GUI provides an intuitive way to modify ligands and save the modified ligand structures in Mol2 format. The resulting structures can be used as input as a parent ligand in the SSFEP Suite. Ligand modifications can be built independently of any other task by choosing *Modify ligand for SSFEP* from the Home page. For instructions on how to modify ligands, please refer to *Ligand Modifications Using the SilcsBio GUI*.

5.1.7 CGenFF Suite Quickstart

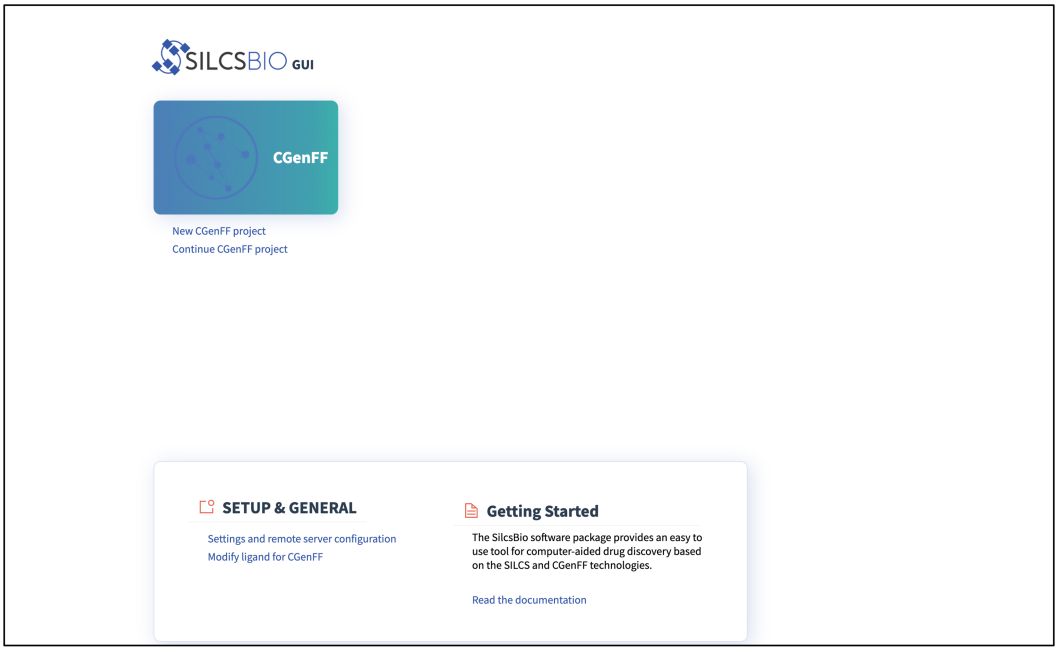
The CHARMM General Force Field (CGenFF) covers a wide range of chemical groups present in biomolecules and drug-like molecules. The CGenFF program automatically assigns high-quality atom types and parameters to an input molecule in seconds. The resulting topology and parameters are compatible with the CHARMM36 force field, allowing simulations of the target molecule with a vast range of macromolecules, including proteins, lipids, nucleic acids, and carbohydrates, as well as other small molecules. The CGenFF program is available through the SilcsBio GUI under the CGenFF Suite. Additional information on CGenFF and the CGenFF program can be found in *Background*.

SilcsBio GUI users can access the CGenFF Suite by selecting *Suites* → *CGenFF* from the menu bar.



The CGenFF Suite Home page provides easy access for users to

- Begin a new CGenFF project (*New CGenFF project*)
- Continue an existing CGenFF project (*Continue CGenFF project*)



The CGenFF Suite also provides convenient access to general tools under *SETUP & GENERAL* at the bottom of the page.

- *Build ligand modifications for CGenFF*
- *Settings and remote server configuration*

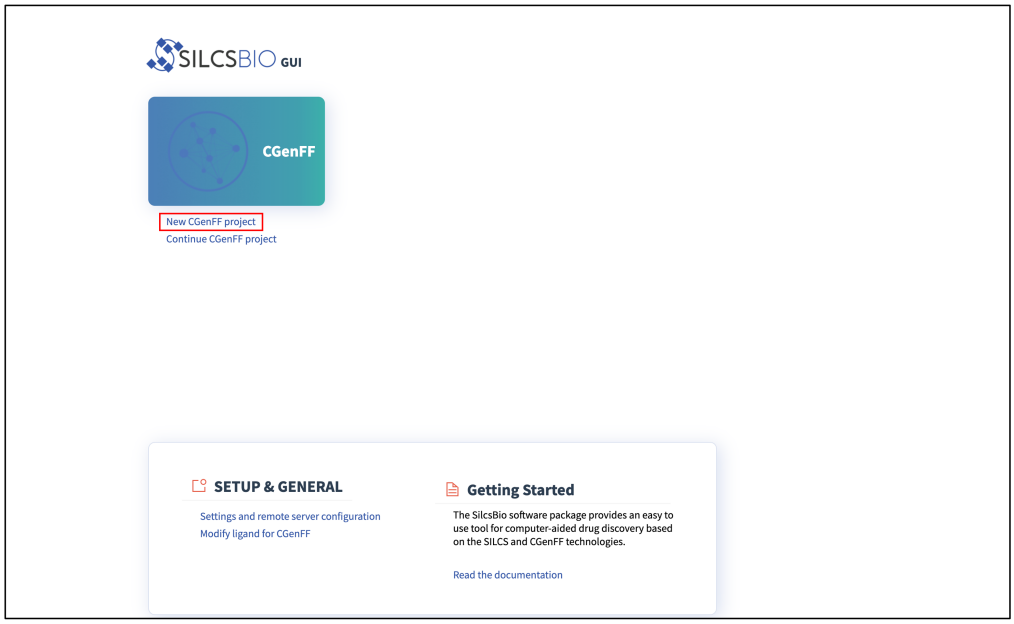
For additional information on these tools, please click on their links. Details on how to use the CGenFF program through the SilcsBio GUI are provided below.

CGenFF Parameter Assignment Using the GUI

To begin a new CGenFF project, follow these steps:

1. **Begin a new CGenFF project:**

   Select *New CGenFF project* from the Home page.



2. **Enter a project name and select the remote server:**

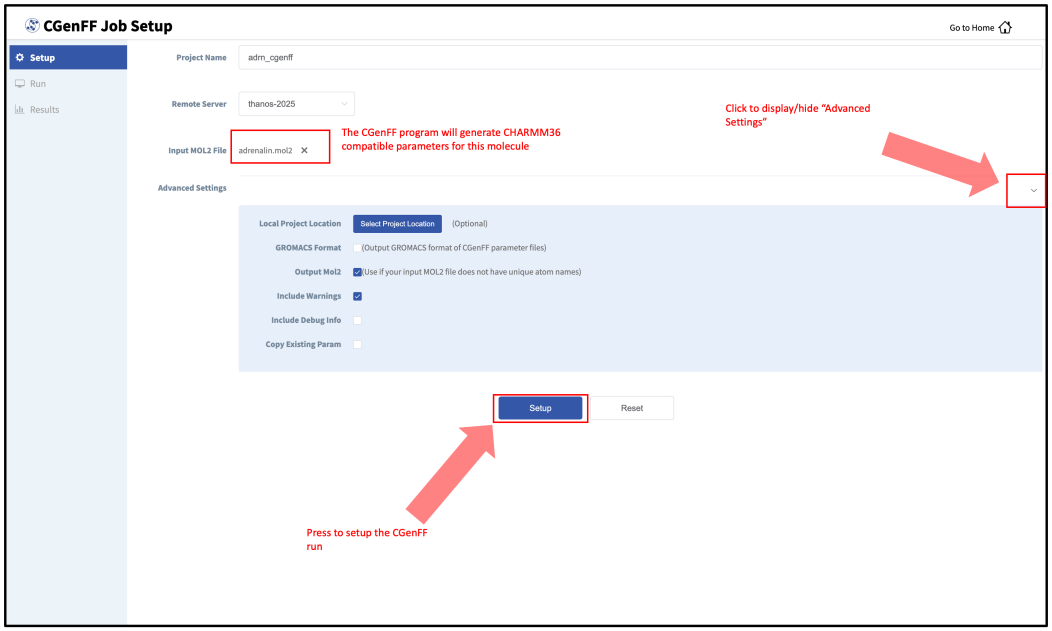
   Enter a project name and select the remote server where the CGenFF job will run. Input and output files from the CGenFF job will be stored on this server. For information on setting up the remote server, please refer to *Remote Server Setup*. Next, select a molecule Mol2 file. As described in *File and Directory Selection*, choose a file from your local computer ("localhost") or from any server you had configured through the *Remote Server Setup* process. For correct atom typing, it is important that all hydrogens are present, the system has correct protonation and tautomeric states, and that the bond orders are correct.

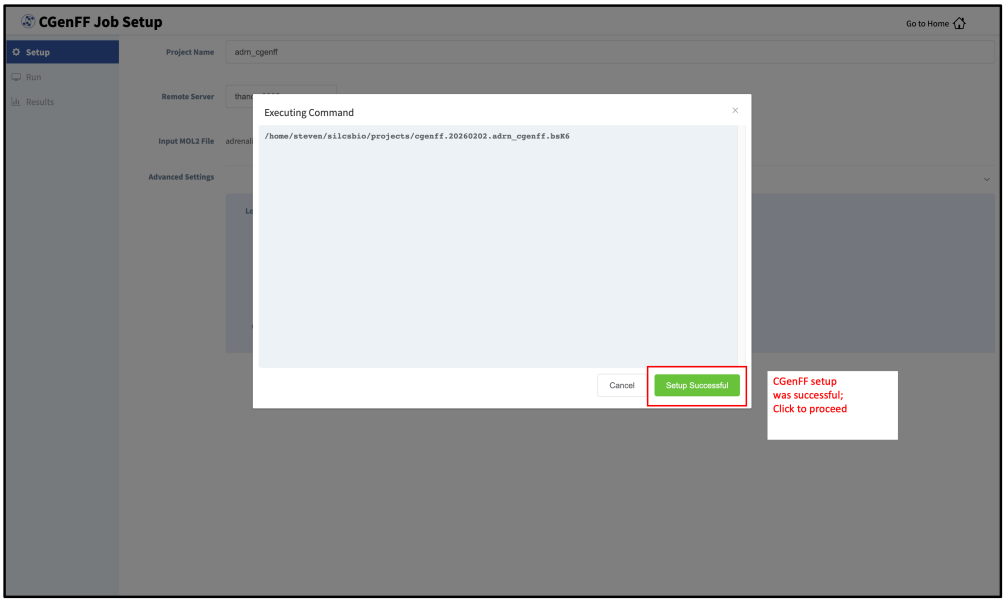
3. **Set up the CGenFF job:**

   Select additional options under "Advanced Settings":

- "GROMACS Format": The CGenFF assigned parameters, by default, are output in CHARMM compatible format. GROMACS compatible topology and parameter files can be specified by selecting the "GROMACS Format" option.
- "Output Mol2": If your input Mol2 does not have unique atom names, the CGenFF program will rename the atoms and the parameter file will reflect the names of the renamed atoms. In this case, it is advised to select the "Output Mol2" option.
- "Include Debug Info": In the event that the input molecule results in an error, you may select the "Include Debug Info" to identify possible erroneous entries in the input Mol2 file (e.g., missing hydrogen atoms).
- "Copy Existing Param": By default, the CGenFF program will only output newly generated parameters as users are expected to use the force field in conjunction with the CHARMM36 force field. The parameters for which CGenFF directly applies to the input molecule can be explicitly listed in the output parameter file by selecting the "Copy Existing Param" option.

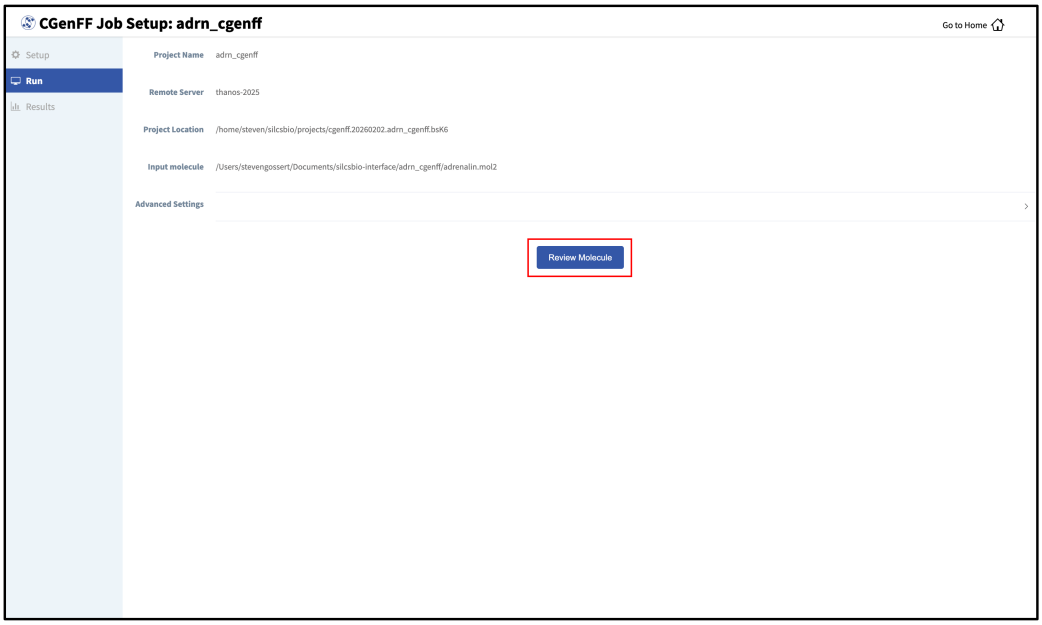
After all information is specified, click the "Setup" button to proceed. Once the CGenFF job is set up, the GUI will indicate that the setup was successful by displaying a green "Setup Successful" button. Click the green "Setup Successful" button to continue.

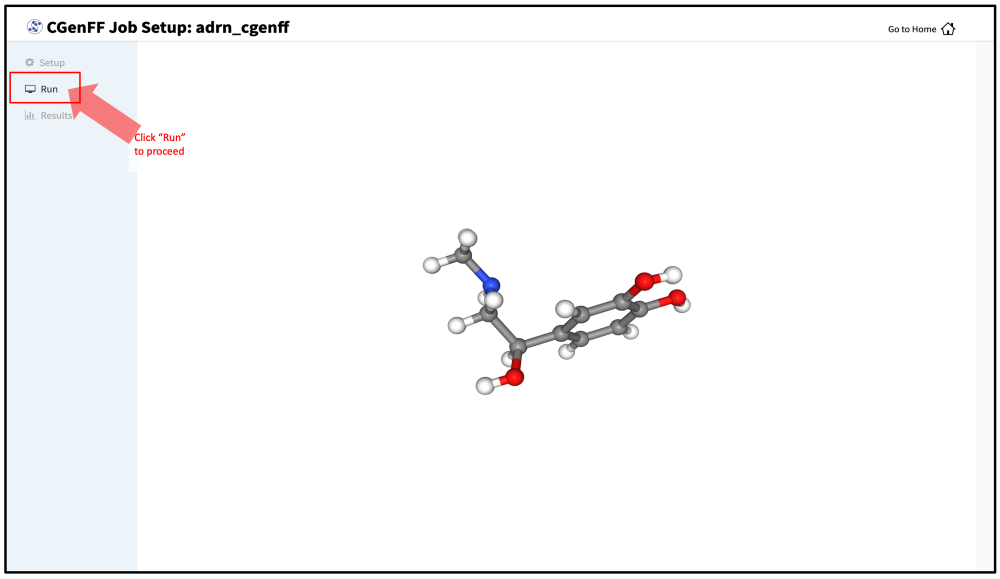




4. **Review the ligand:**

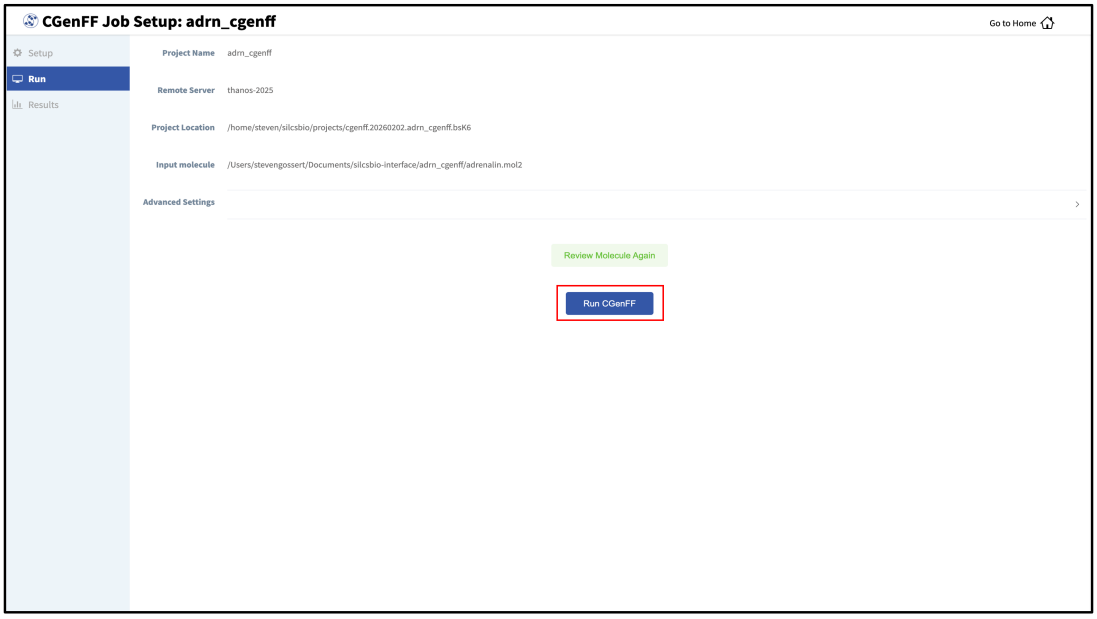
   After setting up the CGenFF run, the GUI will prompt you to review your molecule. Click the "Review Molecule" button at the bottom of the screen to visualize the molecule. After reviewing the molecule, click "Run" in the sidebar menu to continue.





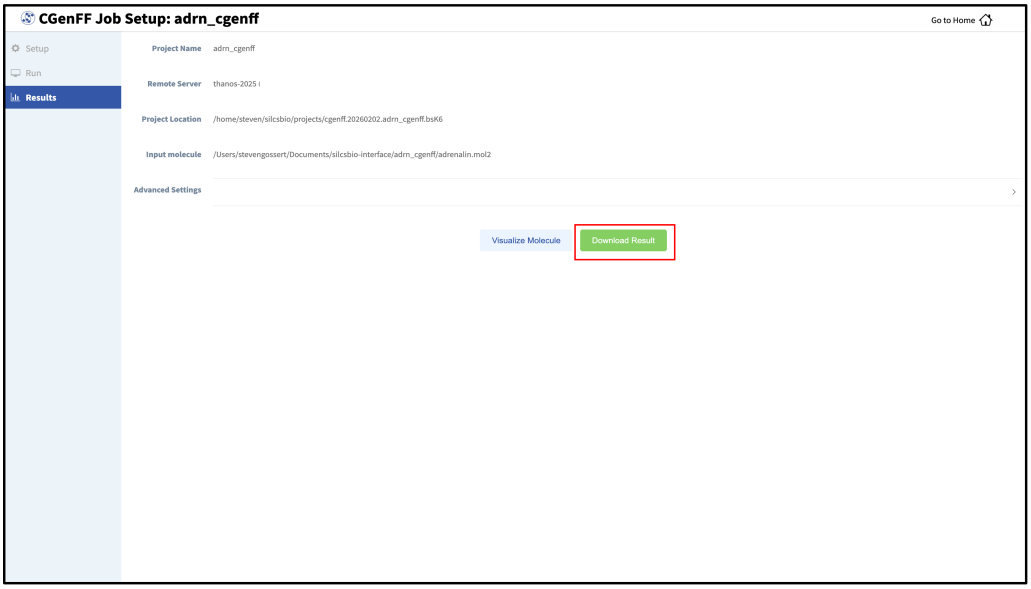
5. **Launch the CGenFF job:**

   To run the CGenFF program, click the "Run CGenFF" button at the bottom of the page. You may also review the molecule again using the "Review Molecule Again" button.



6. **Download CGenFF assigned topology and parameters:**

   Once the CGenFF program assigns force field parameters to the molecule, the results will be downloadable by clicking the green "Download Result" button.



For more information on the output stream file format and advanced CGenFF usage using the CLI, please continue on to *CGenFF Using the CLI*. Please contact support@silcsbio.com if you need additional assistance.

Modify Ligand for CGenFF

Given an input parent ligand structure, the SilcsBio GUI provides an intuitive way to modify ligands and save the modified ligand structures in Mol2 format. The resulting structures can be used as input to the CGenFF program. Ligand modifications can be built independently of any other task by choosing *Modify ligand for CGenFF* from the Home page. For instructions on how to modify ligands, please refer to *Ligand Modifications Using the SilcsBio GUI*.

5.2 Command Line Interface (CLI) Quickstart

This chapter provides step-by-step introductions on how to use the SilcsBio Command Line Interface (CLI). Example commands assume a Bash shell is being used.

5.2.1 SILCS Simulations Using the CLI

1. **Set the environment variables required to access the software:**

```
export GMXDIR=<gromacs/bin>
export SILCSBIODIR=<silcsbio>
```

2. **Set up the SILCS simulations:**

```
${SILCSBIODIR}/silcs/1_setup_silcs_boxes prot=<Protein PDB>
```

   To determine if the setup is complete check that the 10 PDB files required for the simulations are available using the command `ls 1_setup/*_silcs.*.pdb`

3. **Submit the SILCS GCMC/MD jobs to the queue:**

```
${SILCSBIODIR}/silcs/2a_run_gcmd prot=<Protein PDB>
```

   This will submit 10 jobs to the queue. To check job progress, use:

```
${SILCSBIODIR}/silcs/check_progress
```

   This will provide a summary list of the SILCS jobs consisting of the full job path, the job number, the task ID, the job status, and the current/total number of GCMC/MD cycles. Job status values are `Q` for queued, `R` for running, `E` for successfully completed, `F` for failed, and `NA` for not submitted.

4. **Generate FragMaps:**

   When the GCMC/MD jobs are finished, generate FragMaps with:

```
${SILCSBIODIR}/silcs/2b_gen_maps prot=<Protein PDB>
```

   This command will submit 10 jobs to the queue for calculating the occupancy maps from individual runs. Once they are done, run the following command:

```
${SILCSBIODIR}/silcs/2c_fragmap prot=<Protein PDB>
```

   This will create a `silcs_fragmap_<Protein PDB>` folder, which contains the final FragMap files as well as scripts for visualization with external software. Please see *Visualizing SILCS FragMaps* for detailed instructions on visualizing your SILCS FragMaps.

For additional details, please see *SILCS Simulations*.

5.2.2 SSFEP simulations Using the CLI

1. **Set the environment variables required to access the software:**

```
export GMXDIR=<gromacs/bin>
export SILCSBIODIR=<silcsbio>
```

2. **Set up the SSFEP simulations:**

```
${SILCSBIODIR}/ssfep/1_setup_ssfep lig=<Ligand Mol2/SDF file> prot=
↪<Protein PDB>
```

   To determine if the setup is completed check that the PDB files required for the simulations are available using the command `ls 1_setup/*/*_gmx_wat.pdb`. The listing should show PDB files for the ligand alone and for the protein–ligand complex.

3. **Submit the SSFEP MD simulation jobs to the queueing system:**

```
${SILCSBIODIR}/ssfep/2_run_md_ssfep lig=<Ligand Mol2/SDF file> prot=
↪<Protein PDB>
```

   This will submit 10 jobs to the queue, 5 for the protein–ligand complex and 5 for the ligand. To check job progress, use:

```
${SILCSBIODIR}/ssfep/check_progress
```

   This will provide a summary list of the SSFEP jobs consisting of the full job path, the job number, the task ID, the job status, and the current/total number of SSFEP cycles. Job status values and their meanings are listed as follows: Q (queued), R (running), E (successfully completed), F (failed), and NA (not submitted).

4. **Calculate $\Delta\Delta G$ of ligand with selected modifications:**

   When the SSFEP MD simulation jobs are finished, use the ligand modification files to submit the $\Delta\Delta G$ calculations.

```
${SILCSBIODIR}/ssfep/3a_setup_modifications lig=<Ligand Mol2/SDF␣
↪file> prot=<Protein PDB> mod=<modification file>
```

   This command will submit 10 jobs, each of which processes one of the MD trajectories for all modifications in the modification file. Depending on the number and sizes of the modifications, this step may take minutes to several hours to complete.

   Once completed, use the following command to collate the results for all of the modifications:

```
${SILCSBIODIR}/ssfep/3b_calc_ddG_ssfep mod=<modification file>
```

This command will create a `lig_decor.csv` file, which contains the free energy change for each modification relative to the parent ligand. SSFEP is designed to evaluate small modifications and results are best interpreted qualitatively. Therefore it is recommended that only the sign of the change, and not the magnitude, be used to inform decision making: values less than 0 indicate a modification predicted to be favorable.

For additional details, please see *SSFEP: Single Step Free Energy Perturbation*.

CHAPTER

SIX

SMALL MOLECULE SUITE

6.1 FragMap Visualization

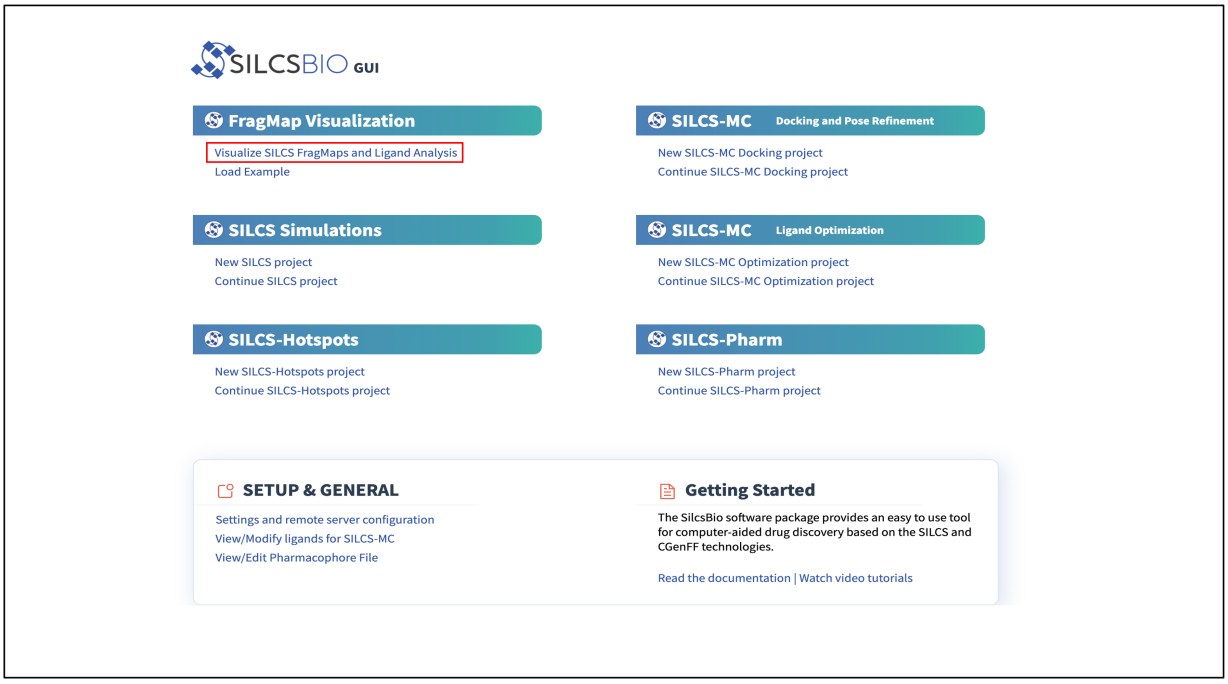
6.1.1 Visualizing SILCS FragMaps

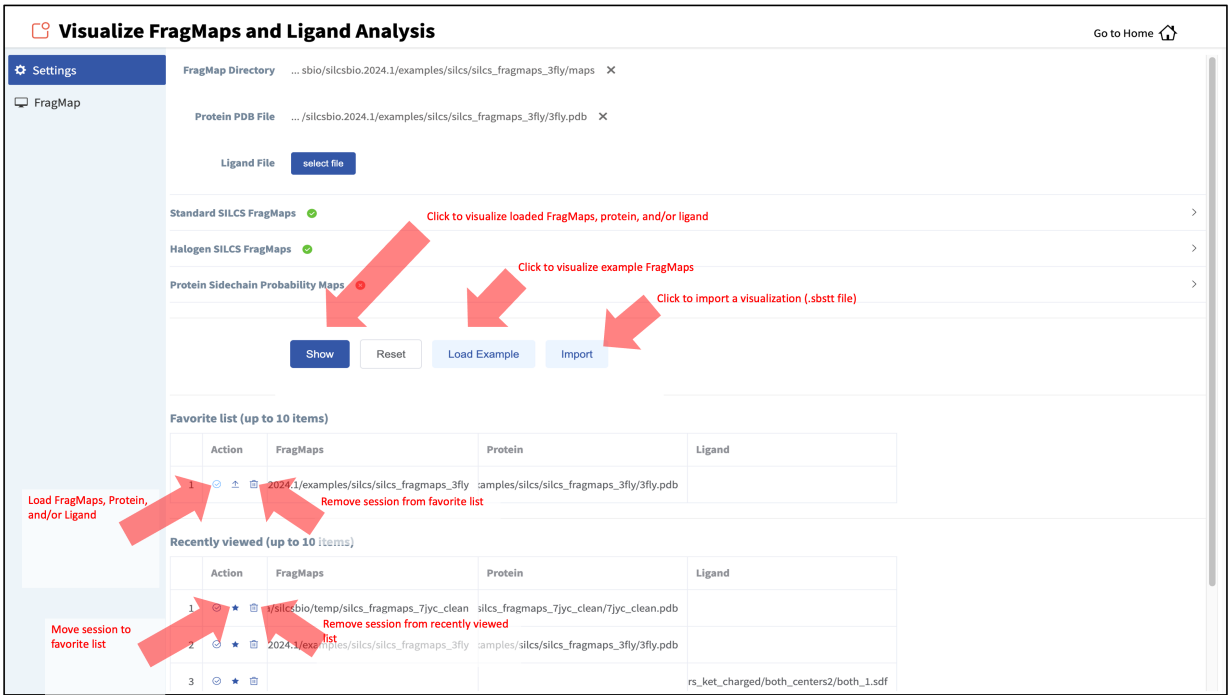
Whether you used the SilcsBIO GUI or the `2c_fragmap` command line utility to create SILCS FragMaps from your SILCS simulation trajectories, you will have a folder `silcs_fragmap_<protein PDB>/maps` that contains SILCS grid free energy (GFE) FragMaps for you system. SILCS FragMaps are defined using non-hydrogen atoms, and for certain solute types (imidazole, formamide) there are multiple FragMaps. FragMaps for water oxygen atoms (tipo) are created to identify water molecules that can be difficult to displace. Generic FragMaps for apolar (benzene+propane), H-bond donor (neutral H-bond donors), and H-bond acceptor (neutral H-bond acceptors) probe types are included to simplify visual analysis.

The easiest way to visualize FragMaps is with the SilcsBio GUI. SilcsBio also provides convenient FragMap visualization using MOE, PyMOL, and VMD.

FragMaps in the SilcsBio GUI

If you have used the SilcsBio GUI to run your SILCS simulations, once the simulations are complete, you will click the "Generate FragMap" button to automatically create FragMaps from the SILCS simulations, download them onto your local computer, and load them, along with the input PDB file, for visualization. Alternatively, you can load SILCS FragMaps from the Home page of the SilcsBio GUI:





If you are loading SILCS FragMaps from the Home page, you will be asked to select your FragMap directory (see *File and Directory Selection*). This directory has a standard name, `silcs_fragmaps_<protein PDB>`, where `<protein PDB>` is the name of the input PDB file. This directory was created on the server where you ran the SILCS jobs either when you clicked the "Generate FragMap" button or when you ran the `${SILCSBIODIR}/silcs/2c_fragmap prot=<protein PDB>` command.

Alternatively, previously viewed visualization sessions may also be loaded by clicking on the checkmark next to the desired session under "Action". The ten most recently viewed visualization sessions will be listed under "Recently viewed". Frequently viewed sessions can be saved under the "Favorite list" by clicking on the star icon next to the session under "Action". Additionally, extraneous sessions can be removed by clicking on the trash icon next to the session under "Action".

Note: In addition to the FragMaps, the `silcs_fragmaps_<protein PDB>` directory contains the PDB file used to run the SILCS simulations. The SilcsBio GUI will automatically detect and load this PDB file. Additionally, if you have run Halogen SILCS, this directory will also contain the Halogen SILCS FragMaps, which will be automatically detected and loaded by the GUI.

Once loaded, the FragMaps and protein will be displayed with default settings. In the default settings, the protein is shown in cartoon representation and generic apolar, generic donor, and generic acceptor FragMaps are shown in green, blue, and red wire representation.

FragMaps can be toggled on and off by clicking the checkbox adjacent to the desired FragMap. Additionally, the FragMap GFE isosurface threshold can be increased or decreased using the + or - buttons. The default FragMap visibilities and FragMap GFE isosurface threshold cutoffs can be restored by clicking on the *Reset to defaults* button. The standard and halogen FragMaps and their associated probe atoms are listed in the tables below.

Table 6.1: Standard SILCS FragMaps

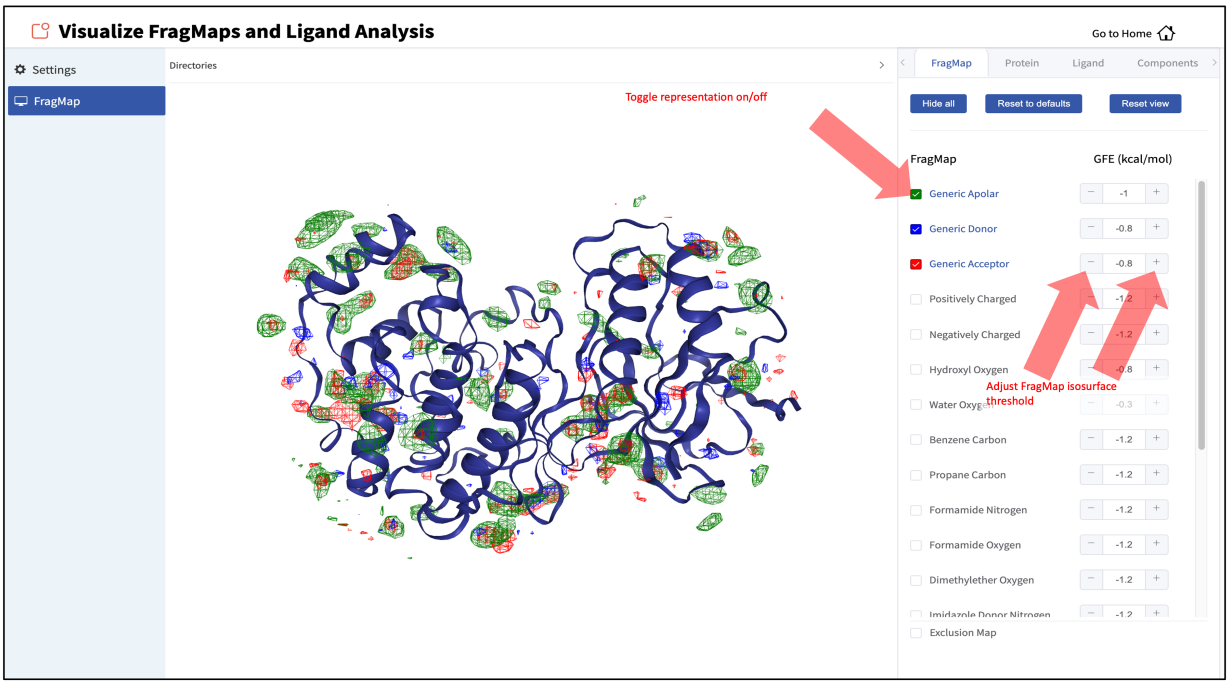
FragMap Name	FragMap Color	Component Name	Associated Probe Atom(s)
Generic Apolar	Green	apolar	Generic Apolar Carbons (benc+prpc)
Generic Donor	Blue	hbdon	Generic Hydrogen Bond Donors (forn+iminh)
Generic Acceptor	Red	hbacc	Generic Hydrogen Bond Acceptors (foro+dmeo+imin)
Positively Charged	Cyan	mamn	Methylamine Nitrogen
Negatively Charged	Orange	acec	Acetate Carboxylate Carbon
Hydroxyl Oxygen	Tan	meoo	Methanol Oxygen
Water Oxygen	Black	tipo	Water Oxygen
Benzene Carbon	Purple	benc	Benzene Carbons
Propane Carbon	Lime Green	prpc	Propane Carbon
Formamide Nitrogen	Blue	forn	Formamide Nitrogen
Formamide Oxygen	Red	foro	Formamide Oxygen
Dimethylether Oxygen	Red	dmeo	Dimethylether Oxygen
Imidizole Donor Nitrogen	Blue	iminh	Imidizole Nitrogen (h-bond donor)
Imidizole Acceptor Nitrogen	Red	imin	Imidizole Nitrogen (h-bond acceptor)

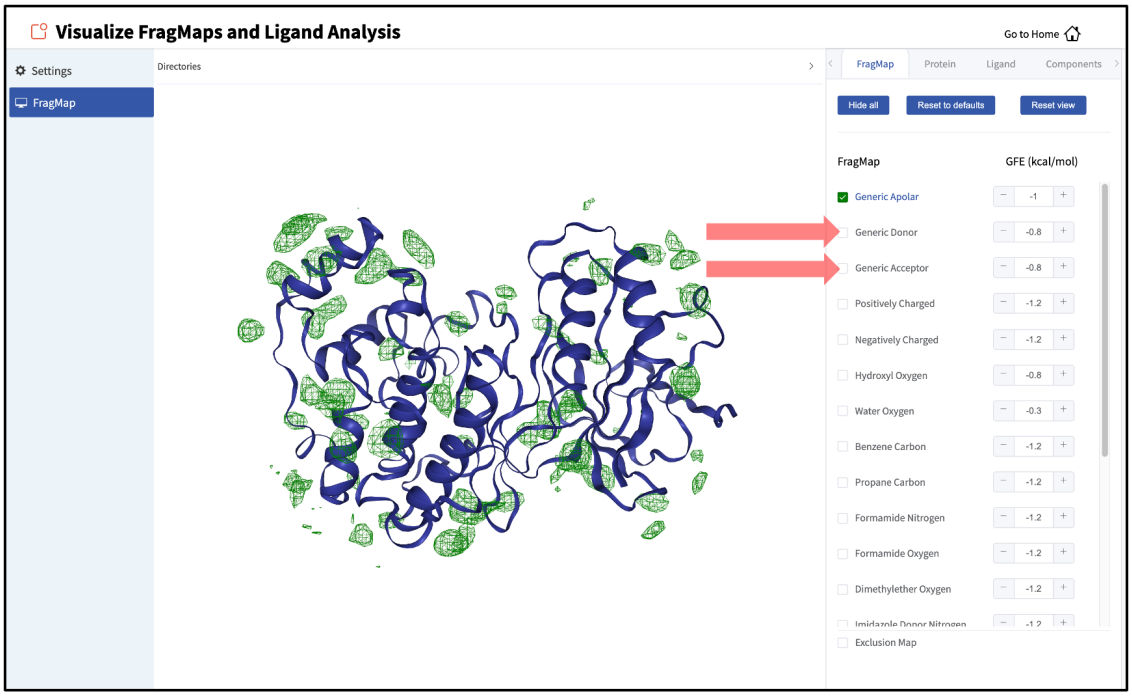
Table 6.2: Halogen SILCS FragMaps

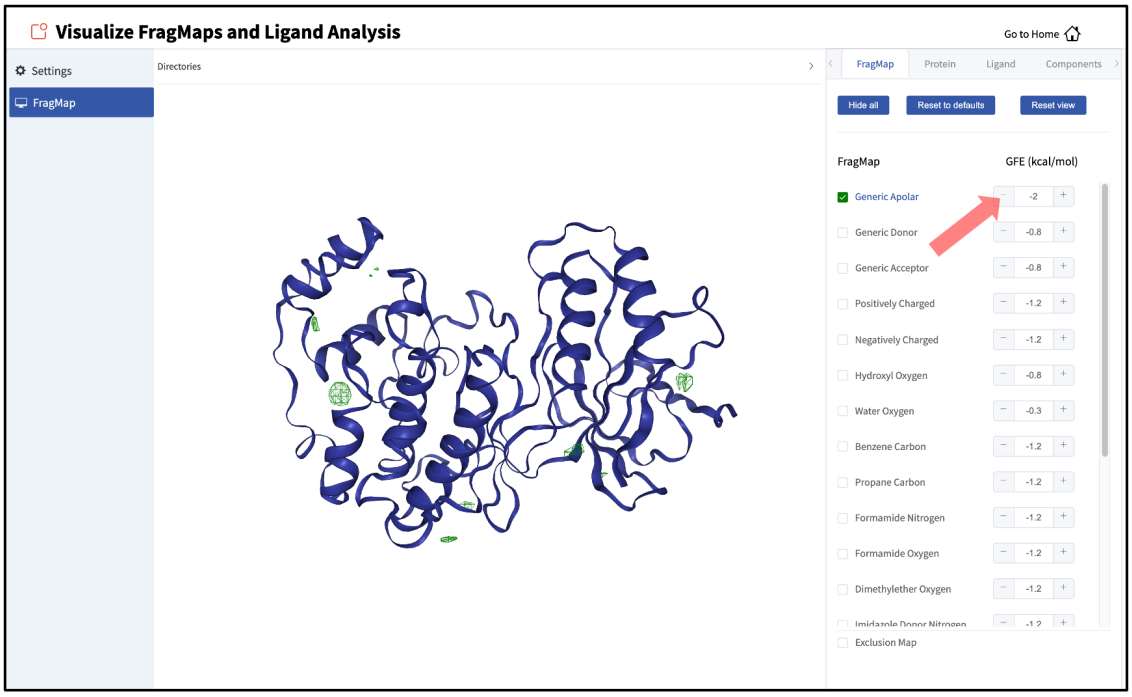
FragMap Name	FragMap Color	Component Name	Associated Probe Atom(s)
Fluorbenzene	Blue	flbc	Fluorbenzene Fluorine
Chlorobenzene	Pink	clbx	Chlorobenzene Chlorine
Bromobenzene	Brown	brbx	Bromobenzene Bromine
Chloroethane	Green	clex	Chloroethane Chlorine
Fluoroethane	Green	fetx	Fluoroethane Fluorine
Trifluoroethane	Green	tfec	Trifluoroethane Carbon Bonded to Fluorines

The FragMap with the name “Exclusion Map” (component name “excl”) is a special case. The SILCS Exclusion Map enumerates all voxels where no probe molecule (including water) sampling

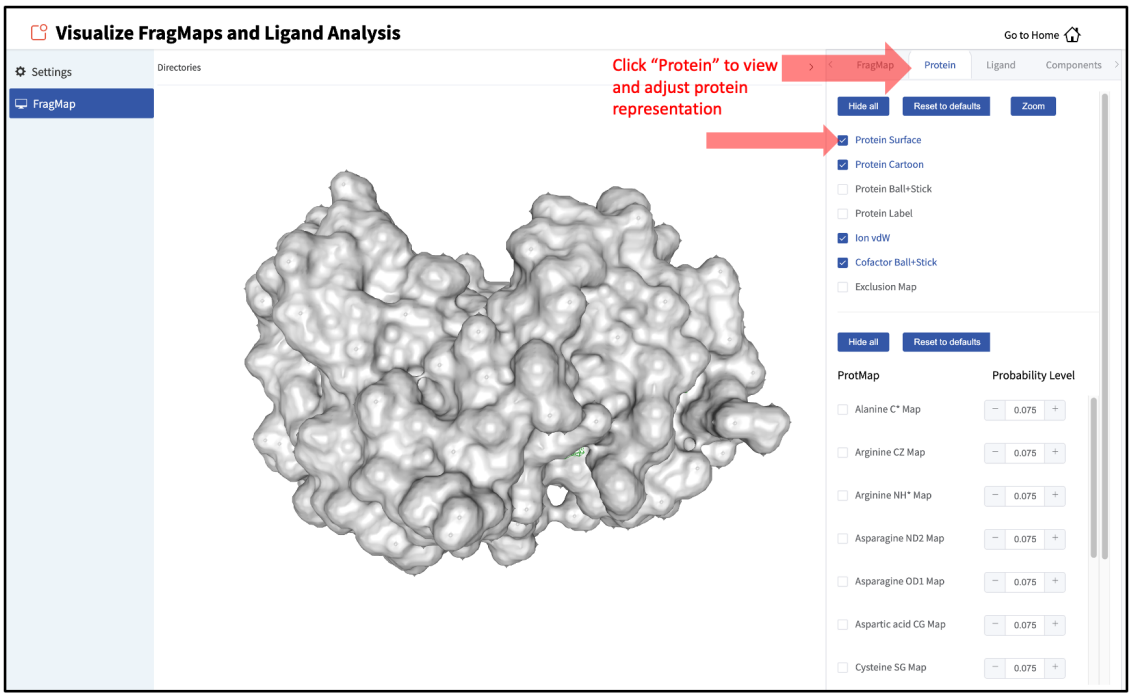
occurred.


Visualize FragMaps and Ligand Analysis
Go to Home
Settings
FragMap
Directories
Toggle representation on/off
FragMap
Protein
Ligand
Components
Hide all
Reset to defaults
Reset view
FragMap
GFE (kcal/mol)
Generic Apolar
Generic Donor
Generic Acceptor
Positively Charged
Negatively Charged
Hydroxyl Oxygen
Water Oxygen
Benzene Carbon
Propane Carbon
Formamide Nitrogen
Formamide Oxygen
Dimethylether Oxygen
Exclusion Map
Adjust FragMap isosurface threshold


Visualize FragMaps and Ligand Analysis
Go to Home
Settings
FragMap
Directories
FragMap
Protein
Ligand
Components
Hide all
Reset to defaults
Reset view
FragMap
GFE (kcal/mol)
Generic Apolar
Generic Donor
Generic Acceptor
Positively Charged
Negatively Charged
Hydroxyl Oxygen
Water Oxygen
Benzene Carbon
Propane Carbon
Formamide Nitrogen
Formamide Oxygen
Dimethylether Oxygen
Exclusion Map

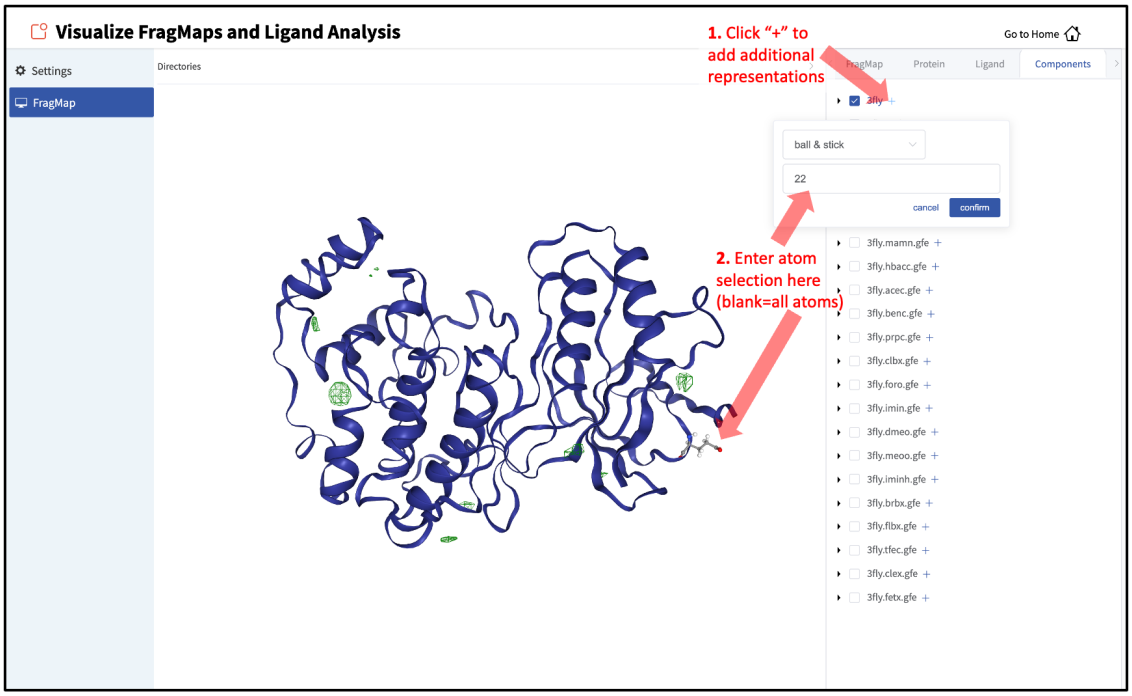


The representation of the protein can also be adjusted by clicking the "Protein" tab of the sidebar menu and toggling on or off the desired protein representations. The default protein representation can be restored by clicking on the *Reset to defaults* button.

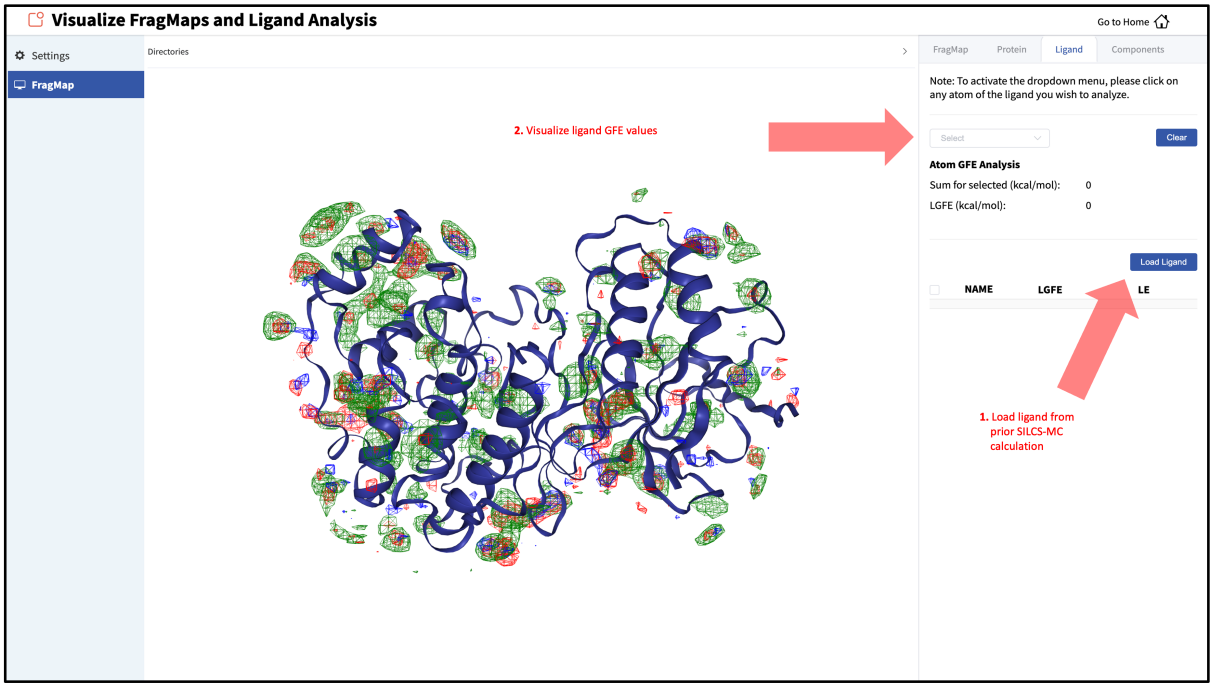


When adding an additional representation, the default is to apply it to all atoms. It is possible to apply it to only a select set of atoms by clicking the "Components" tab of the sidebar menu, clicking the + adjacent to the protein structure name, and typing the atom selection in the blank box above

the “cancel” and “confirm” buttons. The selection syntax is the same as for the NGL molecular structure viewer (see *Atom Selection Syntax*).



It is also possible to load ligands for visualization. If the ligand file was the output of a SILCS-MC calculation, GFE values for the ligand atoms will be embedded within that file. These values can be visualized and their sum automatically computed by first loading the ligand file, and then using the “GFE” tab as detailed at the end of the section *SILCS-MC Ligand Optimization Using the SilcsBio GUI*.



FragMaps in MOE

To visualize your SILCS FragMaps with MOE, first create FragMaps in the MOE-readable CNS format. For this example, if your jobs ran on your server in the directory `~/silcsbio/projects/silcs.5tGP`:

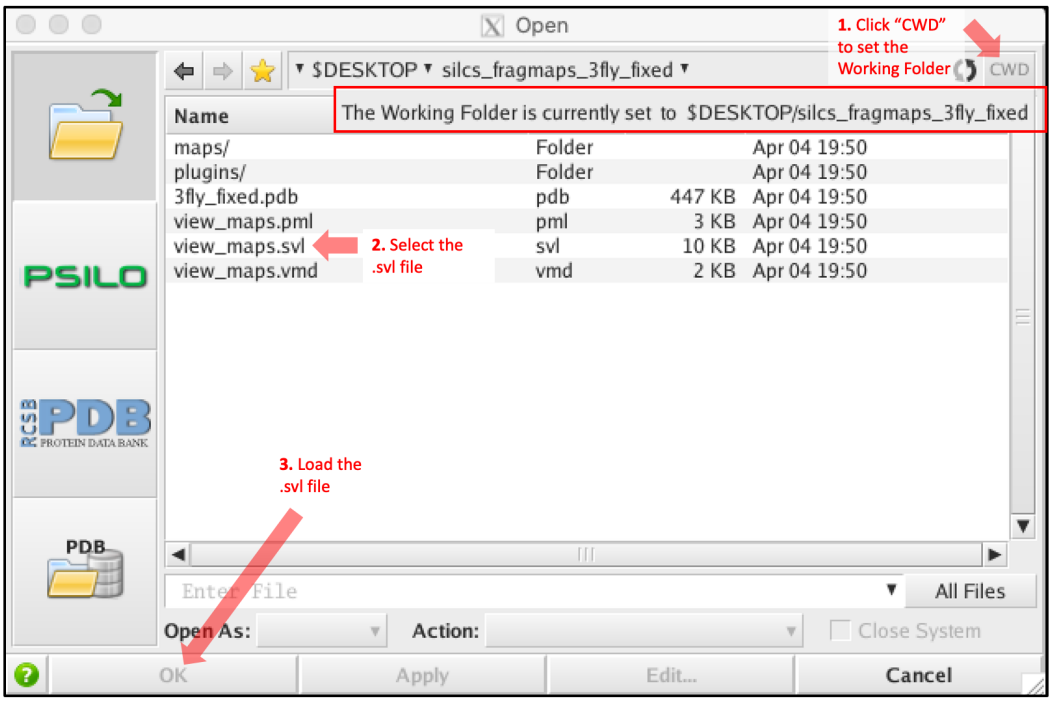
```
cd ~/silcsbio/projects/silcs.5tGP
$SILCSBIODIR/silcs/2c_fragmap prot=<protein PDB> cns=true
```

Note: This step is required ONLY for visualization with MOE. CNS format grid files are approximately 10x the size of the default-format grid files generated and used by your SilcsBio software. PyMol and VMD are capable of reading the default-format files, and, to save disk space, CNS format files are not generated unless specifically specified using the above command.

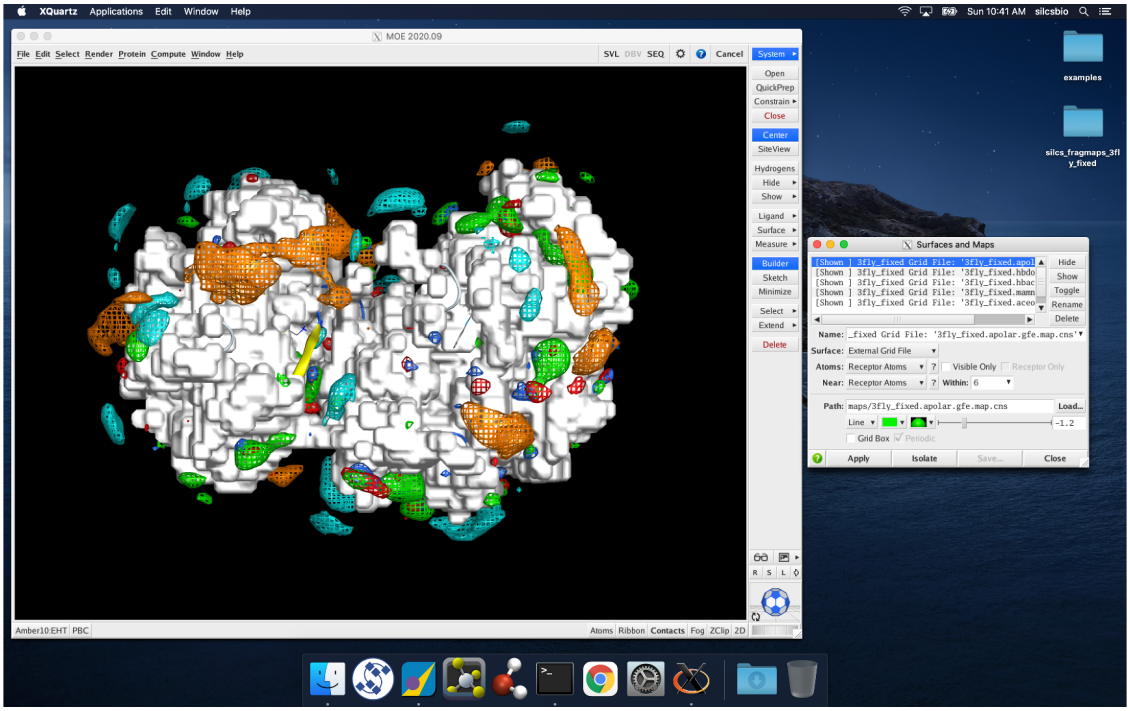
Running this command will create CNS format FragMap files in the `silcs_fragmap_<protein PDB>/maps` directory. It will also create the file `silcs_fragmap_<protein PDB>/view_maps.svl`. First, copy the entire `silcs_fragmap_<protein PDB>` directory from your server to a convenient location on the desktop or laptop machine where you will be doing the visualization. For example, on Linux or MacOS, you may use a command like

```
scp -rp silcsbio@silcsserver:~/silcsbio/projects/silcs.5tGP/silcs_
↪fragmaps_3fly_fixed ~/Desktop
```

Then, use MOE to open the `silcs_fragmap_<protein PDB>/view_maps.svl` file by selecting *File* → *Open*, which will bring up the "Open" window. In this window, browse to the `silcs_fragmap_<protein PDB>` directory, which, in this example, was downloaded from the server to the Desktop of the machine on which we are doing the visualization. Click the "CWD" button in the upper right corner to set this directory as the "Working Folder." Then, select the `view_maps.svl` file and click the "Open" button:



MOE will process the `view_maps.svl` file to load your input PDB structure `<protein PDB>.pdb` used for the SILCS simulations, the CNS format FragMap files `<protein PDB>.<fragmaptype>.map.cns`, and the Exclusion Map `<protein PDB>.excl.map.cns`. The visualization of the FragMaps and Exclusion Map can be controlled in the "Surfaces and Maps" window, which is also automatically loaded by `view_maps.svl`.



FragMaps in PyMol

First, copy the entire `silcs_fragmap_<protein PDB>` directory from your server to a convenient location on the desktop or laptop machine where you will be doing the visualization. For example, on Linux or MacOS, you may use a command like

```
scp -rp silcsbio@silcsserver:~/silcsbio/projects/silcs.5tGP/silcs_
↪fragmaps_3fly_fixed ~/Desktop
```

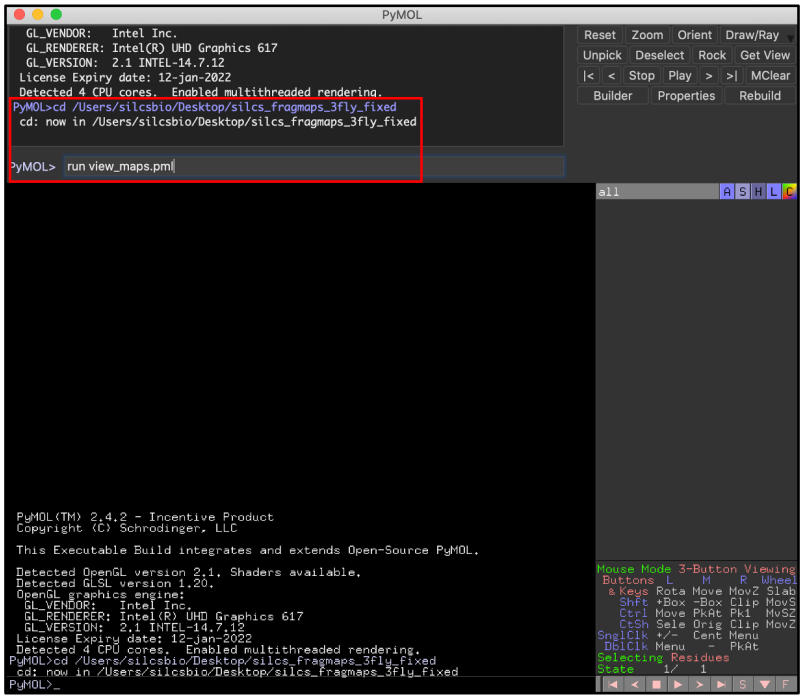
Open PyMol, then select *File* → *Open...*, and open the `view_maps.pml` file located within the `silcs_fragmaps_<protein PDB>` folder. It is also possible to do this by typing commands into the PyMol console. For the `silcs_fragmaps_3fly_fixed` example here, the commands would be

```
cd /Users/silcsbio/Desktop/silcs_fragmaps_3fly_fixed
```

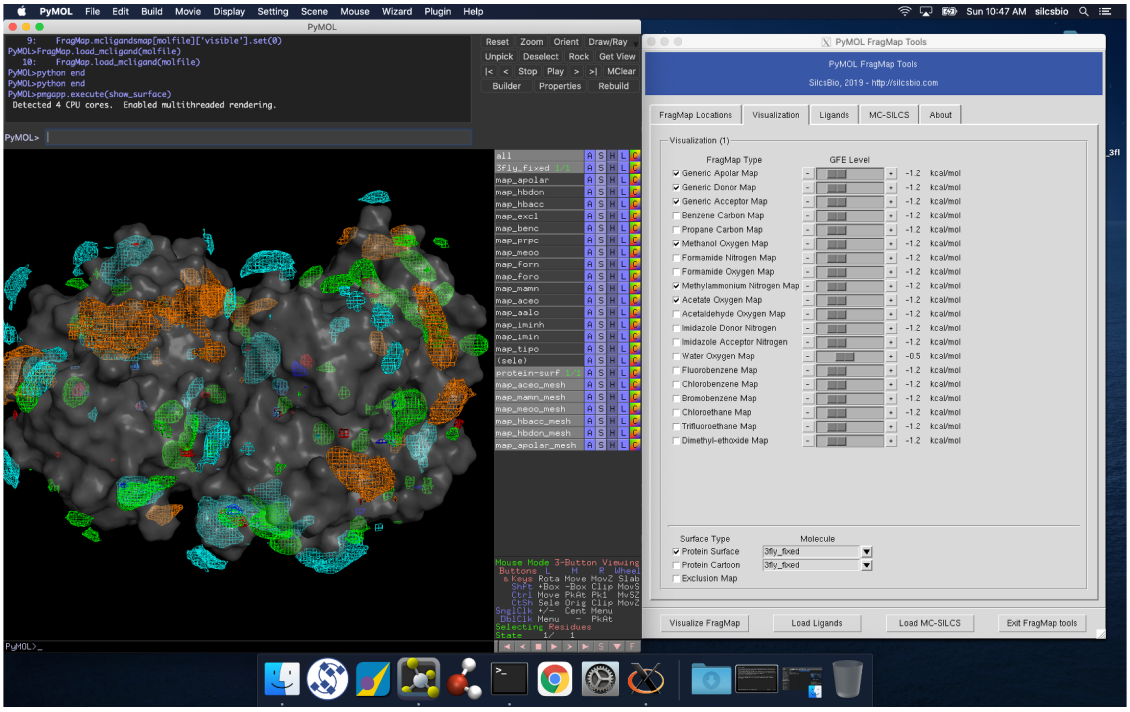
which sets the working directory, and

```
run view_maps.pml
```

which runs the `view_maps.pml` file in that directory.



Whichever of these two methods you use to load `view_maps.pml`, PyMol wil load the PDB structure `<protein PDB>.pdb` used for the SILCS simulations, the FragMap files, and the Exclusion Map. The visualization of the FragMaps and Exclusion Map can be controlled in the "PyMol FragMap Tools" window, which is also automatically loaded by `view_maps.pml`.



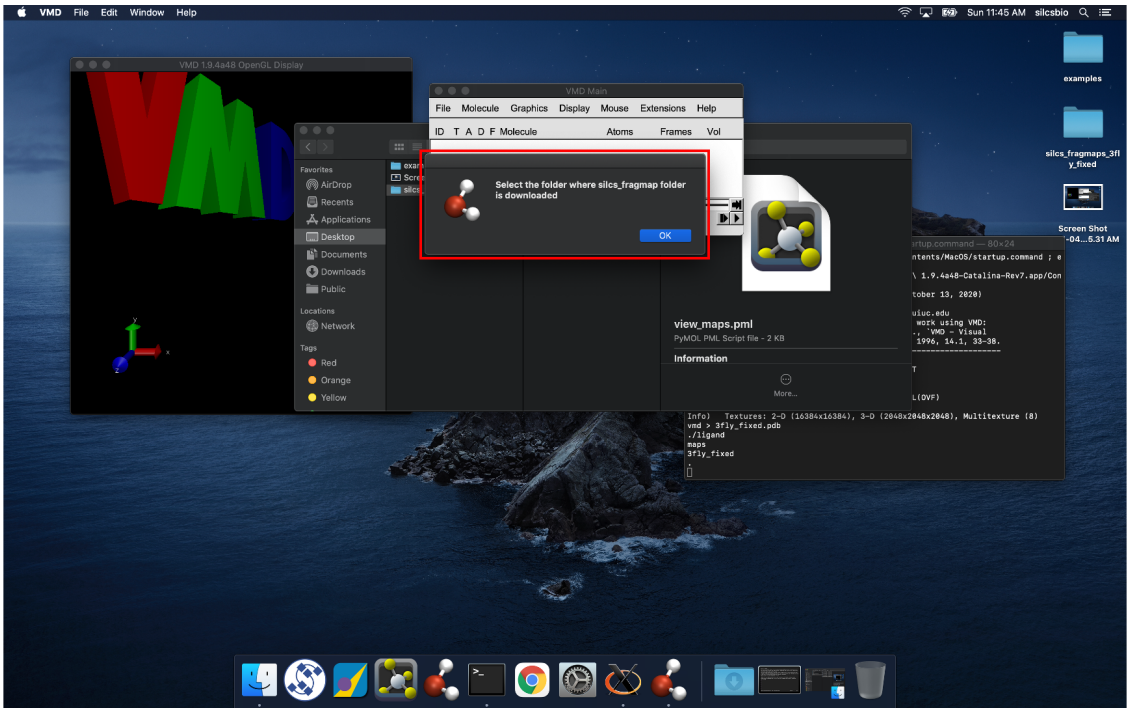
Warning: If your operating system associates the `.pml` extension with PyMol, it is possible simply to double click on the `view_maps.pml` file in the `silcs_fragmaps_<protein PDB>` folder. However, PyMol may fail to open multiple FragMaps. Specifically, if you are opening two different FragMaps having the same protein name, using double clicking to open `view_maps.pml` will result in the first FragMap being loaded again instead of the second one. Doing *File* → *Open...* will direct you to provide the correct path for the second FragMap set.

FragMaps in VMD

First, copy the entire `silcs_fragmap_<protein PDB>` directory from your server to a convenient location on the desktop or laptop machine where you will be doing the visualization. For example, on Linux or MacOS, you may use a command like

```
scp -rp silcsbio@silcsserver:~/silcsbio/projects/silcs.5tGP/silcs_
↪fragmaps_3fly_fixed ~/Desktop
```

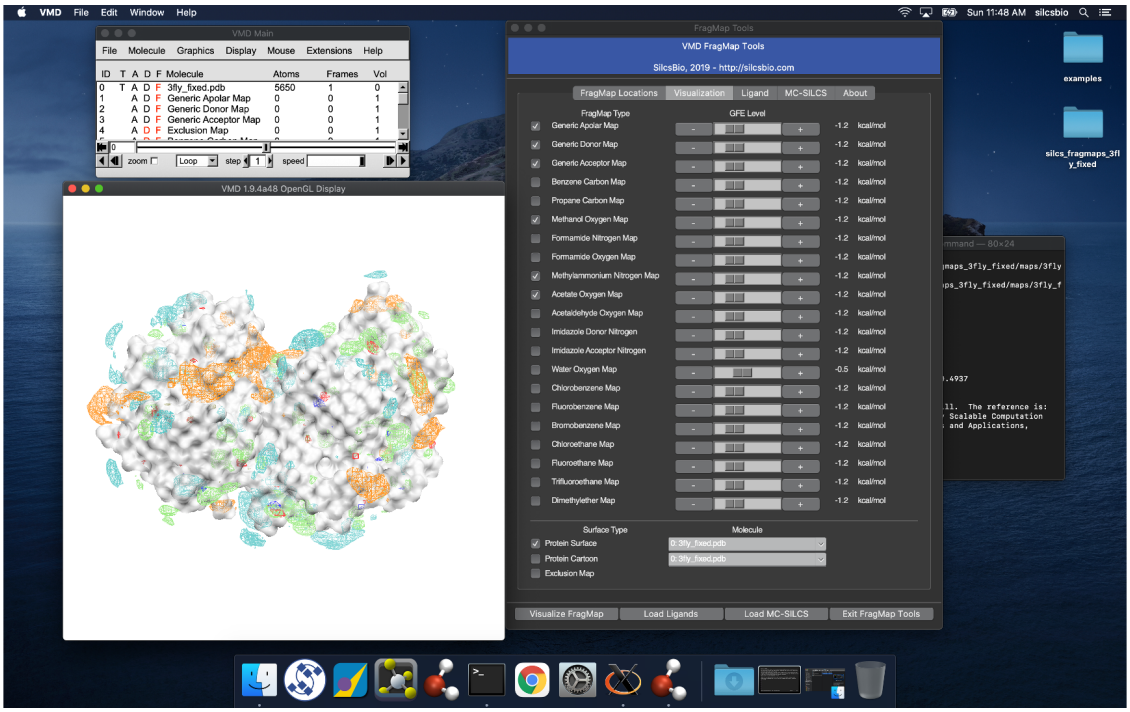
Open VMD, select *File* → *Load Visualization State...*, and open the `view_maps.vmd` file located in the `silcs_fragmap_<protein PDB>` directory you downloaded. A new window will pop up that says, "Select the folder where silcs_fragmap folder is downloaded". Click the "OK" button and select the `silcs_fragmap_<protein PDB>` you downloaded.



Tip: Be careful to select the `silcs_fragmap_<protein PDB>` folder and NOT the

`silcs_fragmap_<protein PDB>/maps` folder.

VMD wil load the PDB structure `<protein PDB>.pdb` used for the SILCS simulations, the FragMap files, and the Exclusion Map. The visualization of the FragMaps and Exclusion Map can be controlled in the "VMD FragMap Tools" window, which is also automatically loaded by `view_maps.vmd`.



Note: VMD (version 1.9.4 or older) does not read .map files correctly. It shifts the grid map by 1.73 angstrom. Visualization in PyMol is preferred.

6.2 SILCS Simulations

6.2.1 Background

Fundamental Idea Behind SILCS

The design of small molecules that bind with optimal specificity and affinity to their biological targets, typically proteins, is based on the idea of complementarity between the functional groups in a small molecule and the binding site of the target. Traditional approaches to ligand identification and optimization often employ the one binding site/one ligand approach. While such an approach might be straightforward to implement, it is limited by the resource-intensive nature of screening

and evaluating the affinity of large numbers of diverse molecules.

Functional group mapping approaches have emerged as an alternative, in which a series of maps for different classes of functional groups encompass the target surface to define the binding requirements of the target. Using these maps, medicinal chemists can focus their efforts on designing small molecules that best match the maps.

Site Identification by Ligand Competitive Saturation (SILCS) offers rigorous free energy evaluation of functional group affinity pattern for the entire 3D space in and around a macromolecule (globular protein [6], membrane protein, or RNA [7]). The SILCS method yields functional group free energy maps, or FragMaps, which are precomputed and then used to rapidly facilitate ligand design. FragMaps are generated by molecular dynamics (MD) simulations that include macromolecule flexibility and explicit solvent/solute representation, thus providing an accurate, detailed, and comprehensive set of data that can be used in database screening, fragment-based drug design, and lead optimization of small molecules.

In the context of biological therapeutics, the comprehensive nature of the FragMaps is of utility for excipient design as all possible binding sites of all possible excipients and buffers can be identified and quantified.

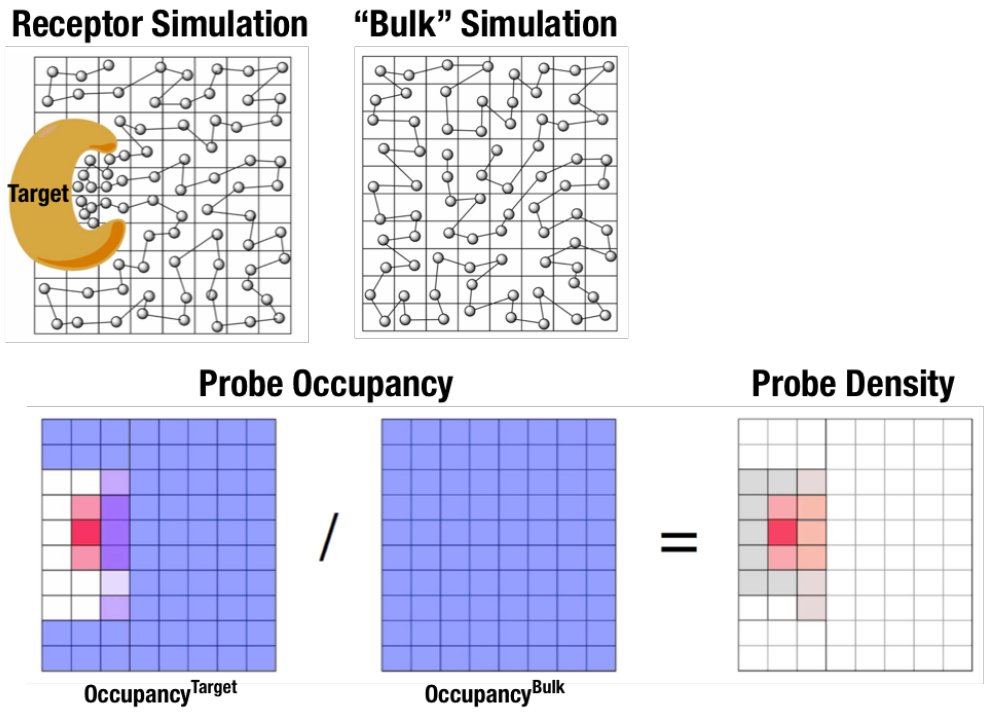


Fig. 6.1: Illustration of fragment density maps.

Fig. 6.1 illustrates FragMaps generation. Two MD simulations of a probe molecule (e.g., benzene) are performed in the presence of a protein and in aqueous solution (without protein), resulting in two occupancy maps, O^{target} and O^{bulk}, respectively. The GFE (grid free energy) FragMaps are

then derived by the following formula [8].

$$\mathrm{GFE}^{T}_{x,y,z} = -RT \log \frac{O^{\mathrm{target}}_{x,y,z}}{\langle O^{\mathrm{bulk}} \rangle}$$

By generating FragMaps using various small solutes (probes) that include a diversity of atoms and chemical functional groups, it is possible to map the functional group affinity pattern of a protein. FragMaps encompass the entire protein such that all FragMap types are in all regions. Thus, information on the affinity of all the different types of functional groups is available in and around the full 3D space of the protein or other target molecule.

A Standard SILCS simulation uses benzene, propane, methanol, imidazole, formamide, dimethylether, methylammonium, acetate, and water as probes to generate Standard SILCS FragMaps. A Halogen SILCS simulation uses fluorobenzene, chlorobenzene, bromobenzene, fluoroethane, chloroethane, and trifluoroethane as probes to generate Halogen SILCS FragMaps. Halogen SILCS FragMaps are used to augment the information provided by Standard SILCS FragMaps.

FragMaps may be used in a qualitative fashion to facilitate ligand design by allowing the medicinal chemist to readily visualize regions where the ligand can be modified or functional groups added to improve affinity and specificity. As the FragMaps include protein flexibility, they indicate regions of the target protein that can "open" thereby identifying regions under the protein surface accessible for ligand binding. In addition to FragMaps, an Exclusion Map is generated based on regions of the system that are not sampled at all by any probe molecules, including water, during the MD simulations. Therefore, the Exclusion Map represents a strictly inaccessible surface.

Many proteins contain binding sites that are partially or totally inaccessible to the surrounding solvent environment that may require partial unfolding of the protein for ligand binding to occur. SILCS sampling of such deep or inaccessible pockets is facilitated by the use of a Grand Canonical Monte Carlo (GCMC) sampling technique in conjunction with MD simulations [9]. Thus, the SILCS technology is especially well-suited for targeting the deep or seemingly inaccessible binding sites found in targets like GPCRs and nuclear receptors.

Once a set of ligand-independent SILCS FragMaps are produced, they can be used for various purposes that rapidly rank ligand binding in a highly computationally efficient manner for multiple ligands **WITHOUT** recalculating the FragMaps. Applications of SILCS methodology include:

- Binding site identification
  - Binding pocket searching with known ligands
  - Binding pocket identification via pharmacophore generation
  - Binding pocket identification via fragment screening
- Database screening
  - SILCS 3D pharmacophore models
  - Ligand posing using available techniques (Catalyst, etc)
  - Ligand ranking

- Fragment-based ligand design
  - Identification of fragment binding sites
  - Estimation of ligand affinity following fragment linking
  - Expansion of fragment types
- Ligand optimization
  - Qualitative ligand optimization by FragMaps visualization
  - Quantitative evaluation of ligand atom contributions to binding
  - Quantitative estimation of relative ligand affinities
  - Quantitative estimation of large numbers of ligand chemical transformations

SILCS generates 3D maps of interaction patterns of functional group with your target molecule, called FragMaps. This unique approach simultaneously uses multiple different small solutes with various functional groups in explicit solvent MD simulations that include target flexibility to yield 3D FragMaps that encompass the delicate balance of target-functional group interactions, target desolvation, functional group desolvation and target flexibility. The FragMaps may then be used to both qualitatively direct ligand design and quantitatively to rapidly evaluate the relative affinities of large numbers of ligands following the precomputation of the FragMaps.

This chapter goes over the workflow for generating SILCS FragMaps.

FragMap names and underlying probe atoms

Each FragMap generated from a SILCS simulation is the result of computing occupancies for one or more kinds of atoms associated with the SILCS probes. The tables below provide the complete lists for Standard SILCS and for Halogen SILCS simulations.

Table 6.3: Standard SILCS FragMaps

FragMap name	generated from probe atom(s)
acec	acetate carboxylate carbon
apolar	generic apolar carbons (benc+prpc)
benc	benzene carbons
dmeo	dimethyl ether oxygen
forn	formamide nitrogen
foro	formamide oxygen
hbacc	generic hydrogen bond acceptors (foro+dmeo+imin)
hbdon	generic hydrogen bond donors (forn+iminh)
imin	imidizole nitrogen (h-bond acceptor)
iminh	imidizole nitrogen (h-bond donor)
mamn	methylamine nitrogen
meoo	methanol oxygen
prpc	propane carbon
tipo	water oxygen

Table 6.4: Halogen SILCS FragMaps

FragMap name	generated from probe atom
brbx	bromobenzene bromine
clbx	chlorobenzene chlorine
clex	chloroethane chlorine
fetx	fluoroethane fluorine
flbc	fluorbenzene fluorine
tfec	trifluoroethane carbon bonded to fluorines

The FragMap with the name “excl” is a special case. It is the SILCS Exclusion Map, and it enumerates all voxels where no probe molecule (including water) sampling occurred.

6.2.2 For Globular Proteins

SILCS Simulations Using the SilcsBio GUI

Please see *SILCS Simulations Using the GUI* as described in *Graphical User Interface (GUI) Quick-start* for a step-by-step description for using the SilcsBio GUI for SILCS simulations.

FragMap generation using SILCS begins with a well-curated protein structure file (without ligand) in PDB file format. We recommend keeping only those protein chains that are necessary for the simulation, removing all unnecessary ligands, renaming non-standard residues, filling in missing atomic positions, and, if desired, modeling in missing loops. Please contact support@silcsbio.com if you need assistance with protein preparation.

A typical SILCS simulation can produce output totaling in excess of 100 GB, so please select a "Project Directory" folder on your server with appropriate free space. The default is determined by the "Project Directory" setting during *Remote Server Setup*.

The setup process automatically builds the topology of the simulation system, creating metal-protein bonds if metal ions are present, rotates side chain orientations to enhance sampling, and places probe molecules around the protein. If missing loops were not modeled into the input PDB file, the amino acids on either side of each missing loop will be automatically capped with neutral functional groups.

Tip: For users with access to both the SILCS-Small Molecule Suite and the CGenFF Suite, running standard MD simulation can be helpful to determine the stability of or refine the protein structure to be used as an initial structure for SILCS simulations. Specifically, this may be very helpful if the user has obtained the protein structure from homology modeling or alphafold. For more information on standard MD simulations with the CGenFF Suite, please refer to *Standard Molecular Dynamics (MD) Simulations*.

During setup, you will need to decide whether to run a Standard SILCS Simulation, Standard + Halogen SILCS Simulations, or a Halogen SILCS Simulation (with another project's Standard SILCS FragMaps). All functionality in the SILCS platform requires FragMaps from a Standard SILCS Simulation. FragMaps from Halogen SILCS Simulations can supplement Standard SILCS FragMaps for *SILCS-MC: Ligand Optimization* and for *SILCS-MC: Docking and Pose Refinement*. Visualization of Halogen SILCS FragMaps can be very useful for qualitatively informing ligand design (see *Visualizing SILCS FragMaps*). See *FragMap names and underlying probe atoms* for a complete listing of both Standard and Halogens SILCS probe molecules and the resulting FragMaps.

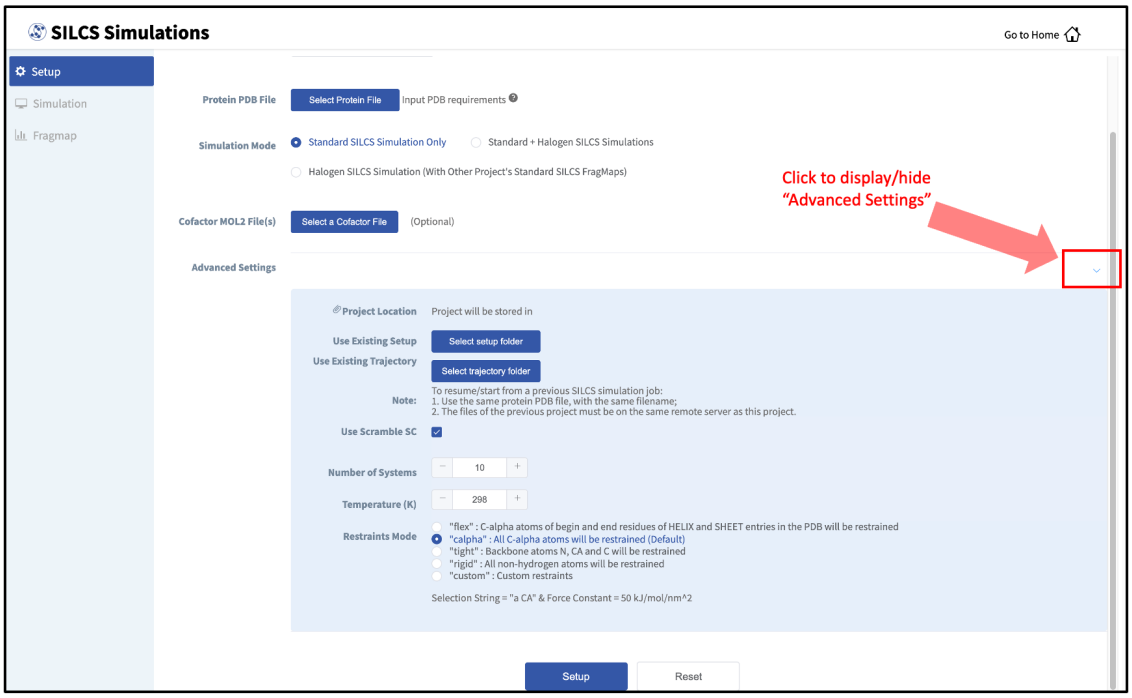
Note: Running a Halogen SILCS simulation requires the same computation time as running a Standard SILCS simulation, so selecting "Standard + Halogen SILCS Simulations" will consume double the computational resources as selecting "Standard SILCS Simulation". The default is to generate and run 10 systems each for a Standard SILCS simulation and for a Halogen SILCS sim-

ulation. Therefore, if your computing resource has 10 nodes, it will take twice as much wall time to complete a "Standard + Halogen SILCS Simulations" job as to complete a "Standard SILCS Simulation". However, if you have access to 20 nodes simultaneously, the wall time will be the same, since each individual system is run independently of the others.

While the same identical input protein coordinates are used for constructing each of these systems, each system is unique because the probe molecules are randomly placed. Additional diversity is added by default to initial coordinates by scrambling the sidechains of exposed amino acids.

Tip: Advanced Settings options during setup allow you to change the scrambling of sidechains (default: on), the number of systems (default: 10), the simulation temperature (default 298 K), and which residues are subjected to restraints on C-alpha atoms (default: all residues). We STRONGLY urge you to use these default settings, which have been selected based on extensive experience.

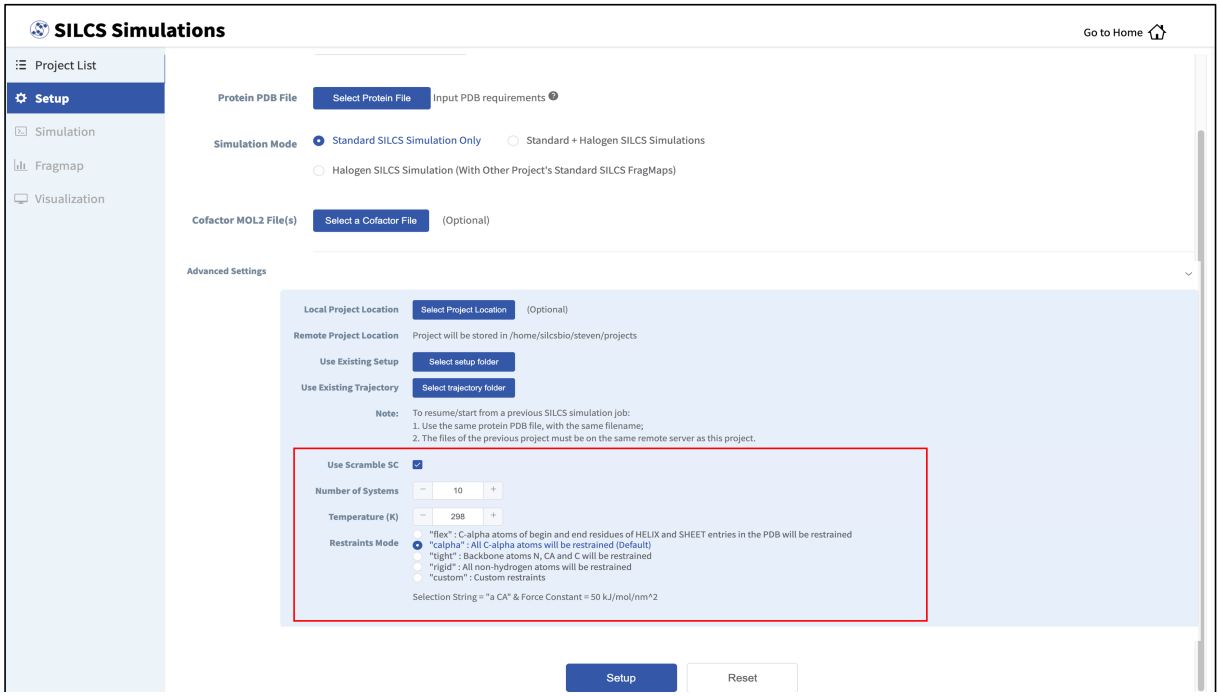
It is also possible to choose an existing setup folder or an existing trajectory folder at this time through the "Advanced Settings" menu.

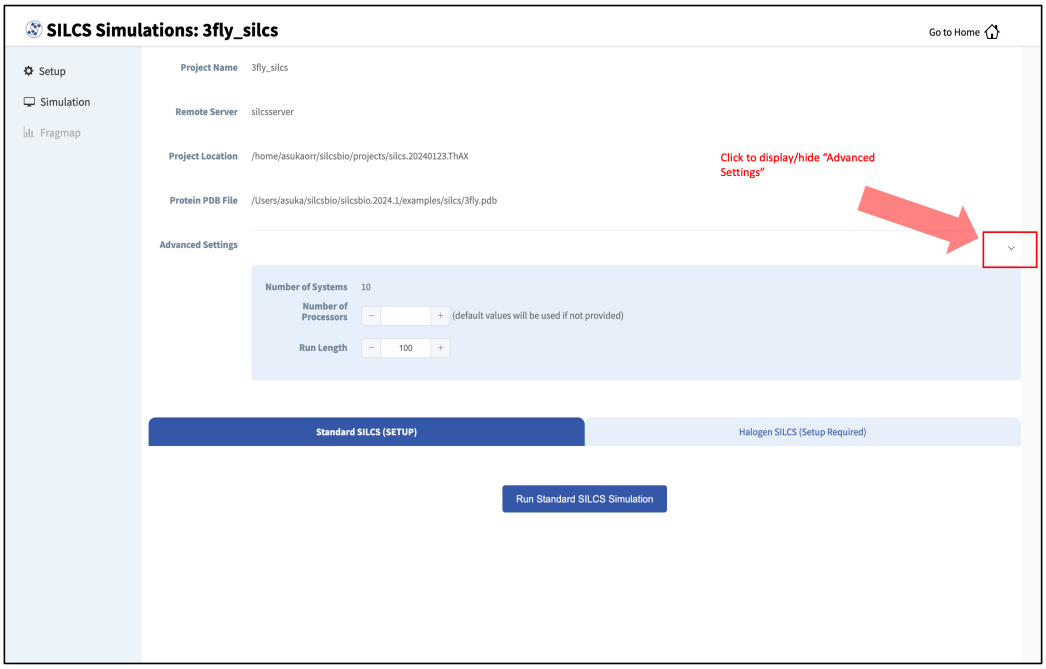


Once the systems are set up, you will need to start the SILCS simulations. There are separate tabs for running the Standard SILCS simulation and the Halogen SILCS simulation, and you will need to use each tab and the "Run…" button underneath that tab to run the corresponding simulation type.

Tip: Advanced Settings options allow you to customize your SILCS simulations. At the "Setup" stage, you may change the number of systems that are run, the simulated temperature, and the

restraints applied to the target macromolecule. After the "Setup" stage and prior to running the simulations, you may change the number of processors used for each run and the length of each run. Leaving these fields blank will result in default values being used. We STRONGLY urge you to use the default settings. If you have concerns about the number of processors per run, please contact your system administrator.





Tip: If a simulation stops prematurely, the progress bar will turn red and show a "Restart" option. When the job is restarted, completed cycles will be automatically skipped and the progress will update to where the job had stopped. That is to say, restarting a failed or stopped job will *not* cause you to lose existing job progress.

SILCS Simulations Using the CLI

Please see *SILCS Simulations Using the CLI* as described in *Command Line Interface (CLI) Quick-start* for a step-by-step description for using the Command Line Interface (CLI) for SILCS simulations. This section covers additional parameters and options that may be included in your SILCS simulations.

FragMap generation using SILCS begins with a well-curated protein structure file (without ligand) in PDB file format. We recommend keeping only those protein chains that are necessary for the simulation, removing all unnecessary ligands, renaming non-standard residues, filling in missing atomic positions, and, if desired, modeling in missing loops. Please contact support@silcsbio.com if you need assistance with protein preparation.

1. **Set up the SILCS simulations:**

```
${SILCSBIODIR}/silcs/1_setup_silcs_boxes prot=<Protein PDB>
```

Warning: The setup program internally uses the GROMACS utility `pdb2gmx`, which may have problems processing the protein PDB file. The most common reasons for errors at this stage involve mismatches between the expected residue name/atom names in the input PDB and those defined in the CHARMM force-field.

To fix this problem: Run the `pdb2gmx` command manually from within the `1_setup` directory for a detailed error message. The setup script already prints out the exact commands you need to run to see this message.

```
cd 1_setup
pdb2gmx -f <PROT PDB NAME>_4pdb2gmx.pdb -ff charmm36 -water␣
↪tip3p -o test.pdb -p test.top -ter -merge all
```

Once all the errors in the input PDB are corrected, rerun the setup script.

The setup command offers a number of options. The full list of options can be reviewed by simply entering the command without any input.

Required parameter:

- Path and name of the input protein PDB file:

```
prot=<protein PDB file>
```

Tip: For users with access to both the SILCS-Small Molecule Suite and the CGenFF Suite, running standard MD simulation can be helpful to determine the stability of or refine the protein structure to be used as an initial structure for SILCS simulations. Specifically, this may be very helpful if the user has obtained the protein structure from homology modeling or alphafold. For more information on standard MD simulations with the CGenFF Suite, please refer to *Standard Molecular Dynamics (MD) Simulations*.

Optional parameters:

- Number of independent simulation systems:

  ```
  numsys=<# of simulations; default=10>
  ```

  SILCS simulations, by default, consist of 10 replicate simulations, each with different initial velocities, to enhance sampling. The number of simulations can be adjusted using the `numsys` keyword.

- Skip the `pdb2gmx` GROMACS utility:

  ```
  skip_pdb2gmx=<true/false; default=false>
  ```

- Path and name of non-covalent ligand/cofactor Mol2 files:

  ```
  lig=<ligand MOL2 files; e.g. "lig1.mol2,/home/johndoe/
  ↪lig2.mol2"; default=empty/NULL>
  ```

  If the input structure contains ligands/cofactors and you wish to compute the SILCS FragMaps in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct position and orientation with respect to the target protein. Provide the path and name of each ligand Mol2 file separated by comma using the `lig` keyword. For covalent ligands, see additional parameters below.

  Note: If your input structure contains a metal ion, please see *What happens when I set up SILCS simulations with an input structure containing a metal ion?*

- Application of weak harmonic positional restraints:

  ```
  restraints_mode=<level of restraints on protein; flex/
  ↪calpha/tight/rigid/custom; default=calpha>
  ```

By default, a weak harmonic positional restraint, with a force constant of ~0.12 kcal/mol/Å^2, is applied to all C-alpha atoms during SILCS simulations. The `restraints_mode` keyword allows the user to customize the restraints applied to the protein.

There are four other options available:

- `flex`

  Selection: C-alpha atoms of begin and end residues of HELIX and SHEET entries in the PDB.

  Force Constant: 50 kJ/mol/nm^2 (~0.12 kcal/mol/Å^2)

  The input PDB file *must* contain HELIX and/or SHEET entries to use the `flex` restraint mode.

- `calpha`

  Selection: C-alpha atoms of all protein residues.

  Force Constant: 50 kJ/mol/nm^2

- `tight`

  Selection: Backbone atoms N, CA and C of all protein residues.

  Force Constant: 100 kJ/mol/nm^2

- `rigid`

  Selection: Non-hydrogen atoms of all residues.

  Force Constant: 150 kJ/mol/nm^2

  By default, using the `rigid` option will turn off the side chain scrambling.

  Tip: It may be beneficial to use the `rigid` option when the binding site is well-known and the protein pdb file is prepared based on a crystal structure of the ligand-bound protein and you wish to optimize the ligand with small modifications. In a retrospective study, the `rigid` option could be used when the user wants to rank a congeneric series of ligands.

- `custom`

  Selection: Custom selection of atoms provided as `restraints_string=` in the command line.

  Force Constant: Custom force constant provided as `restraints_fc=` in the command line.

  For example, if `restraints_string="r 17-160 | r 168-174 & a CA"` and `restraints_fc=1000`, then a force constant of 1000 kJ/mol/nm^2

will be applied to the C-alpha atoms of residues 17 to 160 and residues 168 to 174. Note that the custom selection string should follow the GROMACS selection syntax.

Warning: The `restraints_mode` keyword should be used with caution. SILCS simulations are designed to sample the local conformational space of the protein. Applying highly flexible restraints will result in poor convergence of the occupancies washing the signal from the FragMaps as we need to align the protein conformations to calculate the FragMaps. Also, large translation and rotation of the protein during MD may result in errors during trajectory collection and conversion with `gmx trjconv`.

- Reorientation of residue side chains:

```
scramblesc=<true/false; default=true>
```

  When set to `true` the `scramblesc` keyword reorients the side chain dihedrals of the protein residues. The `rigid` option for the `restraints_mode` keyword will turn off the side chain scrambling.

- Simulation box size:

```
margin=<system margin (Å); default=15>
```

  By default, the simulation system size is determined by adding a 15 Å margin to the size of the input protein structure. The size of the simulation box can be customized by using the `margin` keyword.

- SILCS probes with halogen-containing functional groups:

```
halogen=<true/false; default=false>
```

  SILCS simulations can be performed with SILCS probes that contain halogen-containing functional groups by including `halogen=true` in the command line. See *Setup with halogen probes* for more details.

Covalent Ligand Parameters:

- Add covalent ligands to the system:

```
covalent=<true/false; default=false>
```

  If the input structure contains covalent ligands and you wish to compute the SILCS FragMaps in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct state, position and orientation with respect to the target protein. See *CGenFF for Covalent*

Ligands Using the CLI for more information on preparing covalent ligands for SILCS simulations.

- When covalent=true, provide the path and name of the covalent input file:

```
covalentinp=<covalent input file; default=covalent.
↪inp>
```

Covalent input file format (5 columns separated by space):

<lig-file-path> <linkatom1> <linkatom2> <linkresname> <linkresid>

Where:

- <lig-file-path> = ligand MOL2 file path; e.g., /home/johndoe/lig2.mol2
- <linkatom1> = atom ID for parent atom (integer) [typically a non-hydrogen atom]
- <linkatom2> = atom ID for child atom (integer) [typically a hydrogen atom]
- <linkresname> = residue name for covalent link; CYS/LYS/ASN/SER/THR
- <linkresid> = residue ID for covalent link (integer)

2. **Submit the SILCS GCMC/MD jobs to the queue:**

Following completion of the setup, run 10 GCMC/MD jobs using the following script:

```
${SILCSBIODIR}/silcs/2a_run_gcmd prot=<Protein PDB>
```

This script will submit 10 jobs to the predefined queue. Each job runs out 100 ns of GCMC/MD with the protein and the solute molecules.

This command offers a number of options. The full list of options can be reviewed by simply entering the command without any input.

Required parameter:

- Path and name of the input protein PDB file:

```
prot=<protein PDB file>
```

Optional parameters:

- Use halogen probes:

```
halogen=<use halogen probes; true/false; default=false>
```

The SILCS simulation will be performed with halogen probes when `halogen=true`. For more information, please refer to *Setup with halogen probes*.

- Simulation temperature:

```
temp=<simulation temperature; default=298>
```

  The input simulation temperature should be entered in Kelvin units. By default, the temperature is set to 298 K.

- Number of GCMC/MD cycles:

```
cycles=<number of gcmc/md cycles to run; default=100>
```

  The number of GCMC/MD cycles can be adjusted by using the `cycles` keyword. The first cycle is number 1. The default is 100 cycles.

- Number of cores per simulation:

```
nproc=<# of cores per simulation; default=8>
```

- Generate input files without submitting jobs to the queue:

```
batch<only generate inputs and not submit jobs true/false;
↪ default=false>
```

Tip: SILCS simulations are run at a temperature of 298 K by default, and most published work with SILCS has used this default temperature. It is possible to set a custom temperature by adding the `temp=<simulation temperature; default=298>` option to the `2a_run_gcmd` command. For example, `temp=310` would run the SILCS simulations at 310 K, which may enhance protein conformational sampling.

Once the simulations are complete, the `2a_run_gcmd/[1-10]` directories will contain `*.prod.100.rec.xtc` trajectory files. If these files are not generated, then your simulations are either still running or stopped due to a problem. Look in the log files within these directories to diagnose problems.

To check job progress, use the following command:

```
${SILCSBIODIR}/silcs/check_progress
```

This will provide a summary list of the SILCS jobs consisting of the job path, the job number, the task ID, the job status, and the current/total number of GCMC/MD cycles. Job status values are: Q [queued], R [running], E [successfully ended], F [failed], NA [not submitted]. For progress on SILCS simulations using halogen probes, use `check_progress_x`.

3. **Generate FragMaps:**

   Once your simulations are done, generate the FragMaps using the following command:

```
${SILCSBIODIR}/silcs/2b_gen_maps prot=<Protein PDB>
```

Required parameter:

- Path and name of the input protein PDB file:

  ```
  prot=<protein PDB file>
  ```

Optional parameters:

- Simulation box size:

  ```
  margin=<size of margin in the map; default=10>
  ```

- Generate halogen maps:

  ```
  halogen=<true/false; default=false>
  ```

  If the SILCS simulations were performed with halogen probes, then use `halogen=true`. For more information, please refer to *Setup with halogen probes*.

- Reference structure:

  ```
  ref=<reference PDB; default=same PDB file given in prot>
  ```

  By default, the FragMaps will be oriented around the protein structure that was used as the initial structure for the SILCS simulations. The FragMaps can also be oriented around a different structure by specifying it with `ref=`.

- Generate protein side chain probability maps for each type of residue:

  ```
  protmap=<true/false; generate protein side chain maps;␣
  ↪default=true>
  ```

  The probability distribution of protein side chains can also be calculated and stored in protein side chain maps. When `protmap=true`, this information will be calculated and stored.

- Generate protein side chain probability maps for specific residues:

  ```
  residmap=<generate protmap for specific residues; input␣
  ↪file with "<RESNAME> <RESID>"; default=empty/NULL>
  ```

  The input file should contain two columns separated by a space. The first column should contain the residue name and the second column should contain the residue number e.g., "CYS 105".

- Spacing between voxels:

```
spacing=<map spacing; default=1.0>
```

The probabiliy of each functional group occupying a voxel is used to calculate GFE values. By default, the spacing between each voxel is 1.0 Å.

- First cycle number to use for map generation:

```
begin=<cycle number to begin for map generation;␣
↪default=1>
```

- Last cycle number to use for map generation:

```
end=<cycle number to end for map generation; default=100>
```

- Collect and merge trajectory:

```
traj=<collect and merge protein trajectory; default=true>
```

- Analyze the trajectory for protein dynamics:

```
analyze=<analyze protein trajectory; true/false;␣
↪default=false>
```

When `analyze=true`, the protein trajectory will be analyzed for protein RMSD, RMSF and RGYR. The outputs will be plotted and saved as PNG files in `traj_anlyz` directory. Additional commands will be printed to calculate the Phi-Psi angles and SASA using GROMACS utilities.

- Specification that v2023.x or older was used for `2a_run_gcmd`:

```
oldversion=<true if v2023.x or older versions used to run␣
↪2a_run_gcmd step and if traj=true; default=false>
```

- Rewrite `.xtc` files acording to selection:

```
rewritextc=<input file for gmx make_ndx to rewrite 2a_run_
↪gcmd/*/*nowat.xtc files with new selection;␣
↪default=empty/NULL>
```

The above command will submit 10 single-core jobs that will build occupancy maps (FragMaps) spanning the simulation box for select probe molecule atoms representing different functional groups. For a ~50K atom simulation system, this step takes approximately 10-20 minutes to complete. The FragMaps have a default grid spacing of 1 Å.

The next step is to combine the occupancy FragMaps generated from individual simulations and convert them into GFE FragMaps. Each voxel in a GFE FragMap contains a Grid Free

Energy (GFE) value for the probe type that was used to create the corresponding occupancy FragMap.

```
${SILCSBIODIR}/silcs/2c_fragmap prot=<Protein PDB>
```

Required parameter:

- Path and name of the input protein PDB file:

  ```
  prot=<protein PDB file>
  ```

Optional parameters:

- Generate protein side chain maps:

  ```
  protmap=<true/false; generate protein side chain␣
  ↪probability maps; default=true>
  ```

  The probability distribution of protein side chains can also be calculated and stored in protein side chain maps. When `protmap=true`, this information will be calculated and stored.

- Skip overlap coefficient calculation:

  ```
  skipoc=<true/false; skip overlap coefficient calculation;␣
  ↪default=false>
  ```

- Generate halogen maps:

  ```
  halogen=<true/false; default=false>
  ```

  If the SILCS simulations were performed with halogen probes, then use `halogen=true`. For more information, please refer to *Setup with halogen probes*.

- Generate maps in CNS format:

  ```
  cns=<true/false; generate maps in CNS format;␣
  ↪default=false>
  ```

- First cycle number to use for map generation:

  ```
  begin=<cycle number to begin for map generation;␣
  ↪default=1>
  ```

- Last cycle number to use for map generation:

  ```
  end=<cycle number to end for map generation; default=100>
  ```

GFE FragMaps will be created in the `silcs_fragmap_<protein PDB>/maps` directory, and PyMOL and VMD scripts to load these file will be created in the `silcs_fragmap_<protein PDB>` directory.

Additionally, the convergence of the SILCS simulations will be assessed by calculating the overlap coefficient between the occupancy maps generated from the first and second halves of the runs. The output is written to the `overlap_coeff.dat` file.

4. **Cleanup:**

Raw trajectory output files are large. To reduce disk usage due to these files, this step will delete unnecessary files and create a `.tgz` file for each simulation system under the `2a_run_gcmd/` directory. The command is:

```
${SILCSBIODIR}/silcs/2d_cleanup prot=<Protein PDB>
```

Required parameter:

- Path and name of input protein PDB file:

```
prot=<protein PDB file>
```

Optional parameters:

- Number of simulation systems:

```
numsys=<# of simulations; default=10>
```

- Specification that halogen probes were used in `2a_run_gcmd`:

```
halogen=<true/false; default=false>
```

- Delete original files:

```
delete=<delete original files after tar-balls are created;
↪ true/false; default=true>
```

Only if the tar-balls are created successfully, the original files will be deleted when `delete=true`. Use `delete=false` to keep the original files and delete them later manually.

Setup with halogen probes

SILCS probes with halogen-containing functional groups can be useful for extending the diversity of chemical space covered by FragMaps. While a common approach is to consider halogen atoms as non-polar groups, research on non-covalent "halogen bonding" suggests halogen atoms warrant special treatment.

The CHARMM General Force Field CGenFF v.2.3.0+ allows halogen atoms to be treated with lone-pairs to achieve directionality in polar interactions. These force field improvements are included in SILCS simulations from version 2020.1 of the SilcsBio software. Halogen-protein interactions are determined using a set of halogenated probes in SILCS simulations: chlorobenzene, fluorobenzene, bromobenzene, chloroethane, fluoroethane, and trifluoroethane.

To use these halogenated SILCS probes in your simulation instead of the standard SILCS probes, add the `halogen=true` keyword:

```
${SILCSBIODIR}/silcs/1_setup_silcs_boxes prot=<Protein PDB> halogen=true
${SILCSBIODIR}/silcs/2a_run_gcmd prot=<Protein PDB> halogen=true
${SILCSBIODIR}/silcs/check_progress_x
${SILCSBIODIR}/silcs/2b_gen_maps prot=<Protein PDB> halogen=true
${SILCSBIODIR}/silcs/2c_fragmap prot=<Protein PDB> halogen=true
${SILCSBIODIR}/silcs/2d_cleanup prot=<Protein PDB> halogen=true
```

This will create folders ending with `_x` (i.e., `1_setup_x/`, etc.) to denote SILCS-Halogen simulation systems, as opposed to standard SILCS simulations. The resulting halogen FragMaps will be included in the `silcs_fragmaps_<PROT>/maps` folder. To use halogen FragMaps in SILCS-MC jobs, refer to the `halogen=true` option in *SILCS-MC: Docking and Pose Refinement*.

Warning: SILCS-Halogen FragMaps are intended to complement standard SILCS FragMaps. For analyses using SILCS-Halogen FragMaps, please ensure that standard SILCS FragMaps have also been generated.

Note: For SILCS-membrane, `${SILCSBIODIR}/silcs/1_setup_silcs_boxes` will be replaced with `${SILCSBIODIR}/silcs-memb/1a_fit_protein_in_bilayer` and `${SILCSBIODIR}/silcs-memb/1b_setup_silcs_with_prot_bilayer`.

If you encounter a problem, please contact support@silcsbio.com.

Resuming stopped SILCS jobs

Situations such as exceeding workload manager (job queue) walltime limits can cause SILCS jobs to stop before running to completion. Resuming such jobs from where they stopped is straightforward. On the server where the job was running, go to the directory `<Project Location>/2a_run_gcmd/i`, where `i` is an integer from 1 to 10 and corresponds to the stopped job among the ten SILCS GCMC/MD jobs that were being run for that particular target. In that directory, directly use the file `job_mc_md.cmd` to submit the job to the workload manager, for example `qsub job_mc_md.cmd` for Sun Grid Engine (SGE) and `sbatch job_mc_md.cmd` for Slurm.

To ensure a successful restart, make sure to issue any required extra setup commands (for example, as listed in the "Extra setup" field under *Settings and remote server configuration* from the SilcsBio GUI Home screen) such as `export LD_LIBRARY_PATH=<path/to/cuda/libraries>` or `module load <name of cuda module>` before submitting the job.

6.2.3 For Membrane Proteins

SILCS-Membrane Simulations Using the SilcsBio GUI

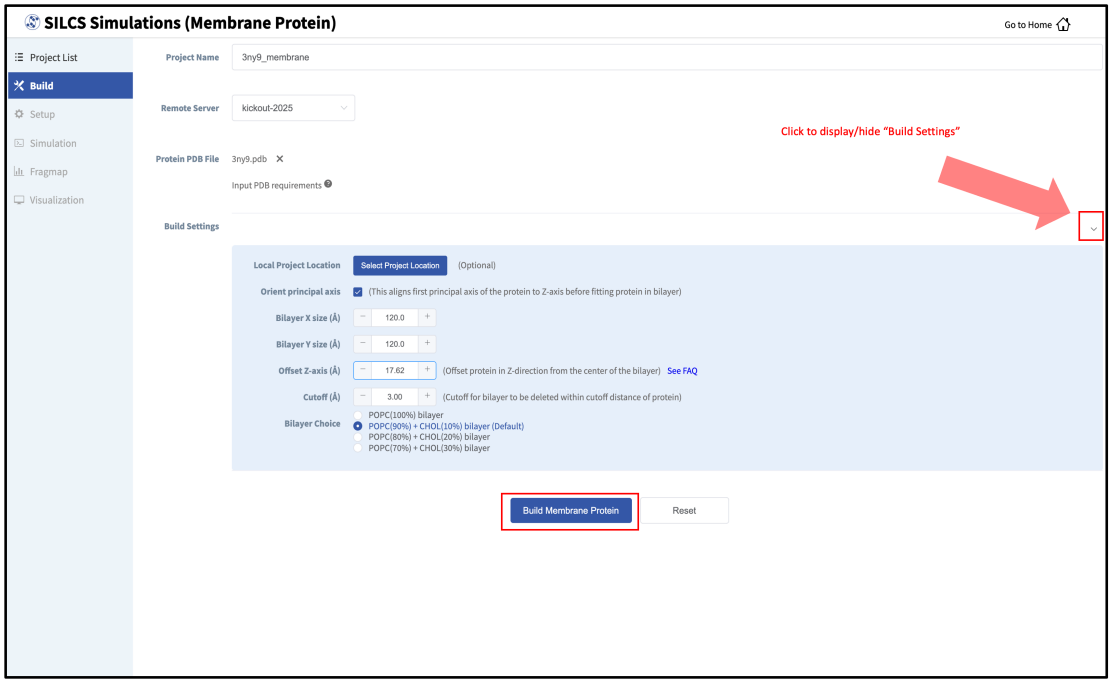
Running SILCS-Membrane using the SilcsBio GUI requires a bare protein structure for best results. SilcsBio provides a utility to embed transmembrane proteins, such as GPCRs, in a bilayer of 9:1 POPC:cholesterol for subsequent SILCS simulations. If you wish to use your own pre-built protein–bilayer system with SILCS, please run SILCS with the CLI.

1. **Begin a New SILCS-Membrane project:**

   To begin a new SILCS-Membrane project, select *New SILCS project* from the Home page as described in *SILCS Simulations Using the GUI* and select "Membrane Protein".

2. **Enter a project name, select the remote server, and upload input files:**

   Enter a project name and select the remote server where the SILCS-Membrane simulation jobs will run. Additionally, provide the protein input file. You may choose the protein input file from the computer where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection*.

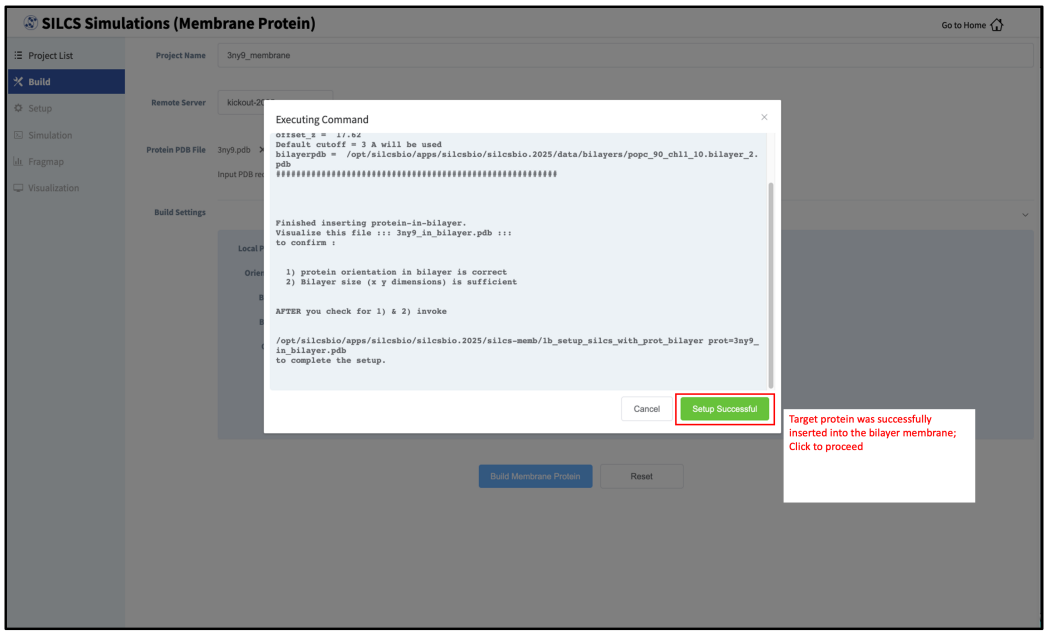


3. **Build the membrane protein system:**

   Once all information is entered correctly, press the "Build Membrane Protein" button at the bottom of the page. This will execute the utility to build the protein/bilayer system. By default, this utility will embed the protein in a bilayer consisting of 9:1 POPC:chelesterol, with the first principal axis of the protein aligned with the bilayer normal (Z-axis) and the protein center of mass translated to the center of the bilayer (Z=0). Additionally, the default system size is set to 120 Å along the X and Y dimensions. These settings can be changed under "Build Settings".

   After the protein is inserted into the bilayer, the resulting protein/bilayer system will be output in a PDB file with suffix `_popc_chol`, and the page will update. Click on the green "Setup Successful" button to return to the GUI.

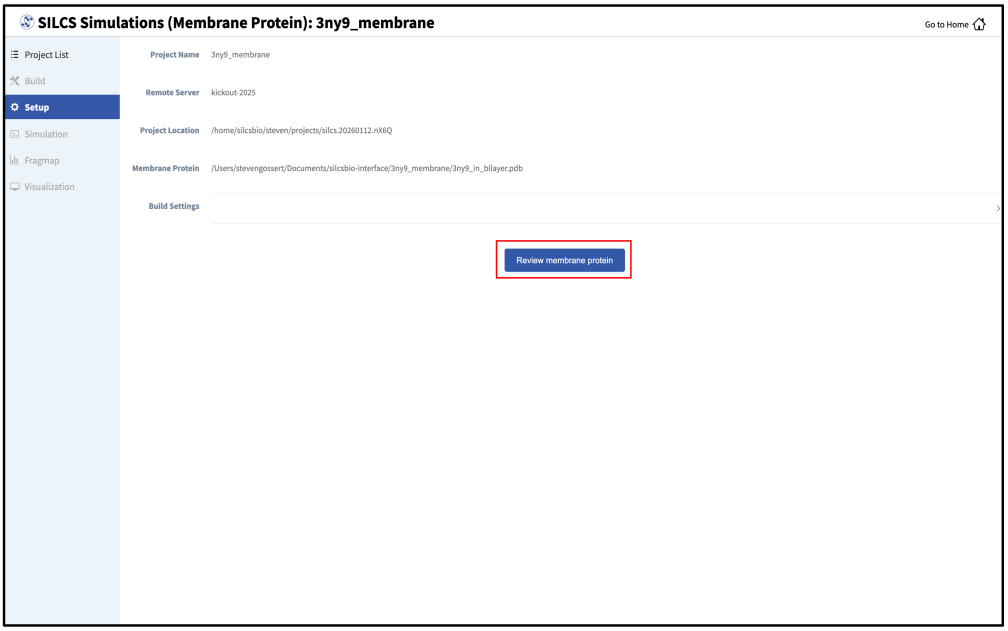
Tip: Build Settings options during setup allow you to change the alignment of the first principal axis of the protein to the Z-axis before fitting the protein to the bilayer (default: on), the bilayer size in the X dimension (default 120 Å), the bilayer size in the Y dimension (default 120 Å), and the offset of the relative position of the protein with respect to the position of the bilayer (default 0 Å). The offset of the protein may be selected in accordance with a recommendation from an external program such as the OPM server. Please see *How do I fit my membrane protein in a bilayer as suggested by the OPM server?* for additional information.

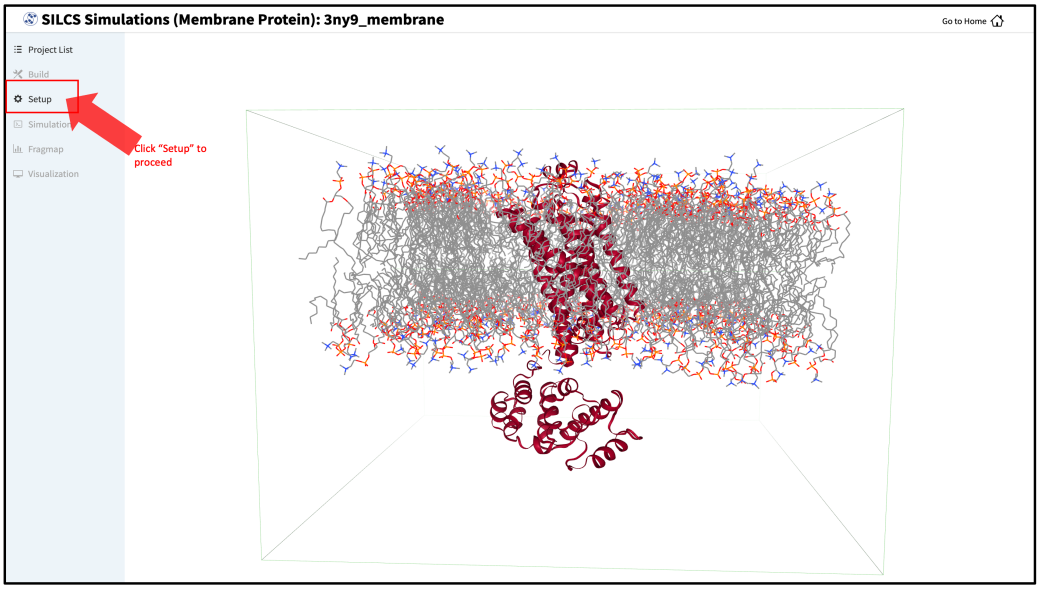


4. **Review the built membrane protein:**

   After the protein is inserted into the bilayer, the GUI will prompt you to review the membrane protein. Click the "Review membrane protein" button to proceed.

   The GUI will then show you the modeled protein/bilayer system through a visualization window. Please confirm that the protein/bilayer system is correct. After reviewing the membrane protein, click "Setup" to proceed





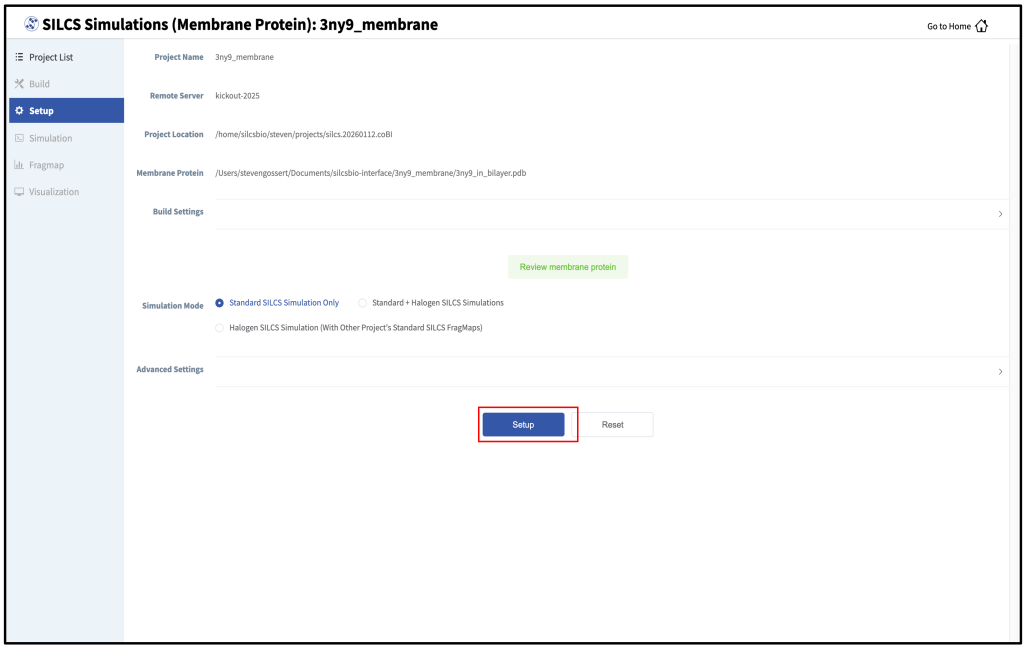
5. **Set up the SILCS simulations:**

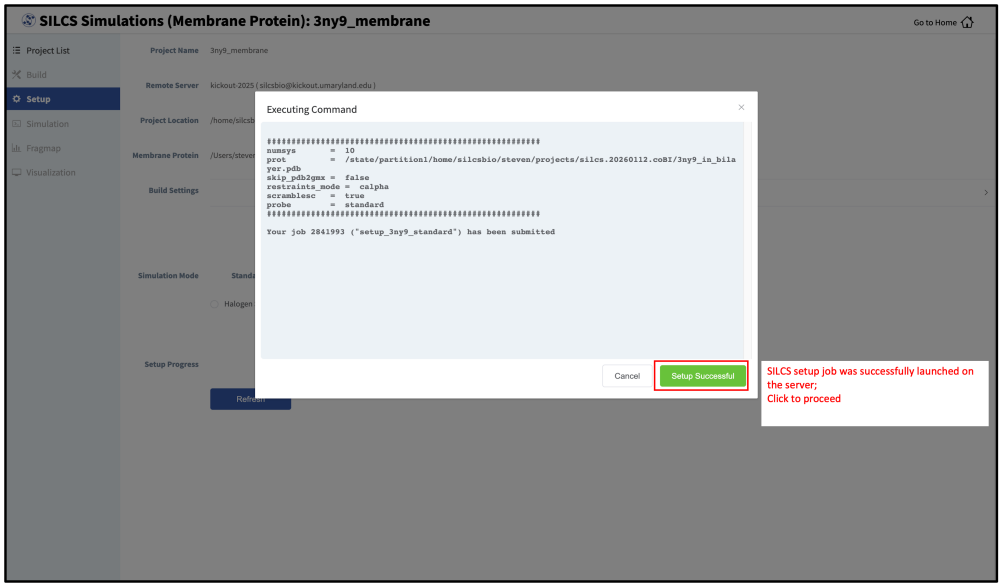
   After reviewing the protein/bilayer system, the GUI will allow you to finalize setting up the SILCS simulations. During this step, you will need to decide whether to run a Standard SILCS Simulation, Standard + Halogen SILCS Simulations, or a Halogen SILCS Simulation (with another project's Standard SILCS FragMaps). All functionality in the SILCS platform requires FragMaps from a Standard SILCS Simulation. FragMaps from Halogen SILCS Simulations can supplement Standard SILCS FragMaps for *SILCS-MC: Ligand Optimization* and for *SILCS-MC: Docking and Pose Refinement*. Visualization of Halogen SILCS FragMaps can be very useful for qualitatively informing ligand design (see *Visualizing SILCS FragMaps*). See *FragMap names and underlying probe atoms* for a complete listing of both Standard and Halogens SILCS probe molecules and the resulting FragMaps.

   Note: Running a Halogen SILCS simulation requires the same computation time as running a Standard SILCS simulation, so selecting "Standard + Halogen SILCS Simulations" will consume double the computational resources as selecting "Standard SILCS Simulation". The default is to generate and run 10 systems each for a Standard SILCS simulation and for a Halogen SILCS simulation. Therefore, if your computing resource has 10 nodes, it will take twice as much wall time to complete a "Standard + Halogen SILCS Simulations" job as to complete a "Standard SILCS Simulation". However, if you have access to 20 nodes simultaneously, the wall time will be the same, since each individual system is run independently of the others.

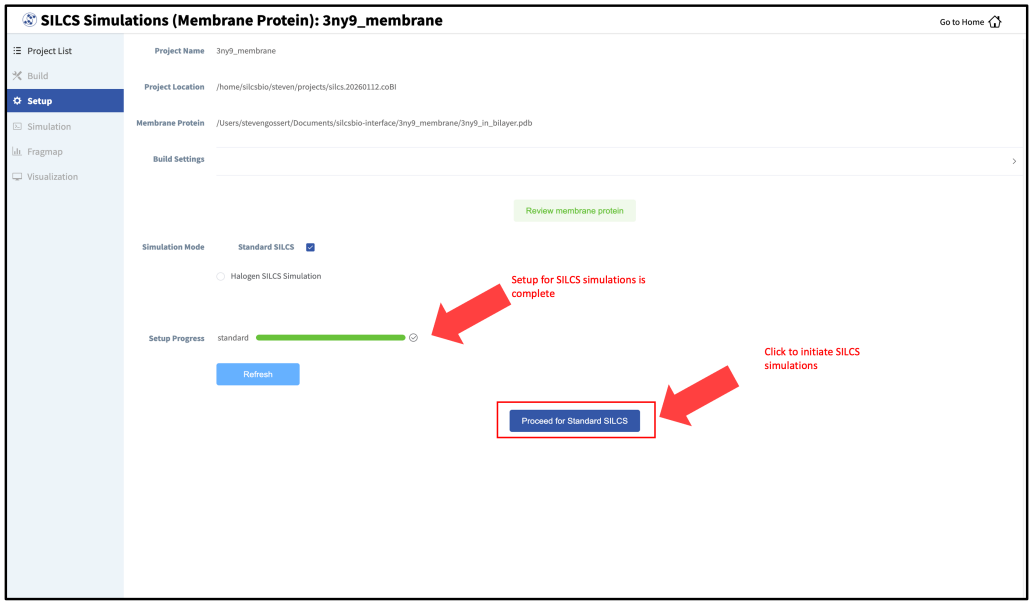
   To proceed to preparing the simulation box, click "Setup". This will submit a job on the server to prepare the SILCS simulation box by adding water and probe molecules to the protein/bilayer system. The GUI will indicate if the job was successfully launched on the

server with a “Setup Successful” button. Click the “Setup Successful” button to proceed.





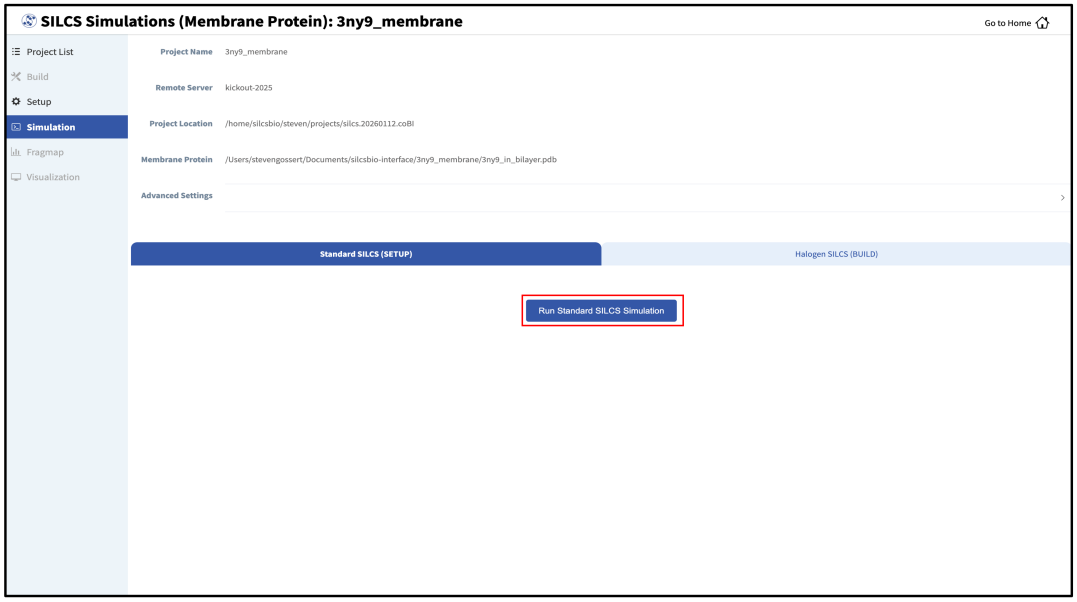
The SILCS simulation setup progress from the server will be displayed on the GUI. Once the SILCS simulation box is successfully set up, and the job is complete, a green checkmark will appear next to the progress bar. Upon completion of the SILCS simulation setup, click the “Proceed for Standard SILCS” button to continue.

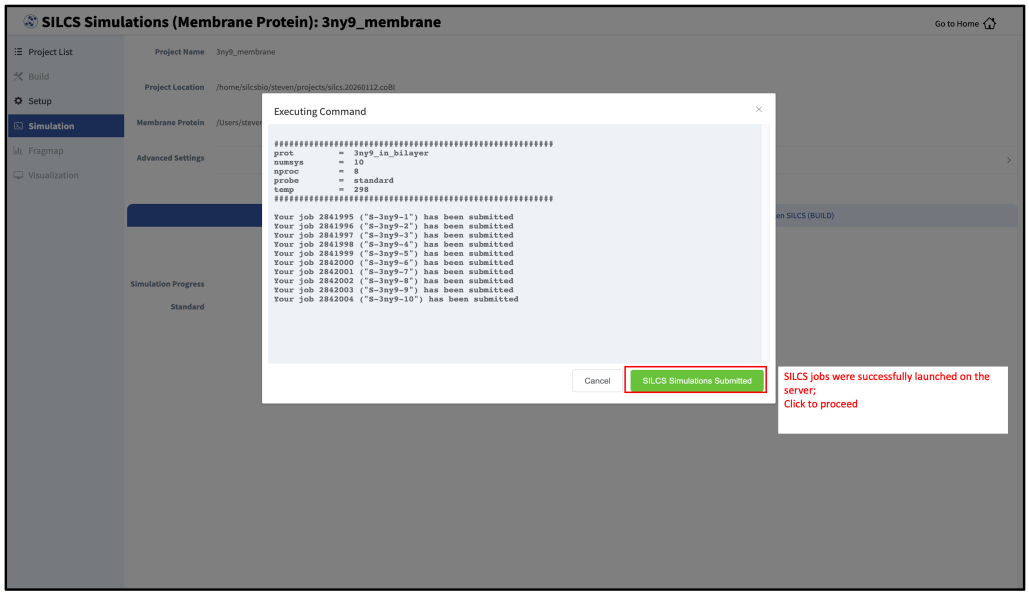


6. **Launch SILCS simulations:**

   The SILCS GCMC/MD simulations can now be launched by clicking the "Run Standard SILCS Simulation" button. The SILCS GCMC/MD will begin with a 6-step pre-equilibration to slowly relax the bilayer followed by 10 ns of relaxation of the protein into the bilayer prior to the production simulation. The GUI will indicate when the SILCS jobs are successfully launched on the server with a green "SILCS Simulations Submitted" button. At this point, the jobs will continue running on the server, and it is safe to exit the GUI.

Tip: Membrane protein simulation systems are typically large and may require significant storage space (> 200 GB) and simulation time (>14 days with GPU).





FragMap generation from the GCMC/MD simulations data follows the same protocol as for non-bilayer systems (i.e., for globular proteins). Refer to previous instructions as described in *SILCS Simulations Using the GUI*.

If you encounter a problem, please contact support@silcsbio.com.

SILCS-Membrane Simulations Using the CLI

1. **Prepare the input protein/bilayer system:**

   You may use either a bare protein structure or your own pre-built protein/bilayer system with SILCS.

   If you are beginning with a bare protein, SilcsBio provides a command line utility to embed transmembrane proteins, such as GPCRs, in a bilayer membrane for subsequent SILCS simulations:

```
${SILCSBIODIR}/silcs-memb/1a_fit_protein_in_bilayer prot=<Protein␣
↪PDB>
```

   This command will align the first principal axis of the protein with the bilayer normal (Z-axis) and translate the protein center of mass to the center of the bilayer (Z=0). The full list of parameters can be reviewed by running the command without any arguments.

   Required parameter:

   - Path and name of the input PDB file:

```
prot=<protein PDB file>
```

   Optional parameters:

   - Apply principal axis-based alignment:

```
orient_principal_axis=<true/false; default=true>
```

     When set to true, first principal axis of the protein is aligned to Z-axis before fitting the protein in bilayer. If the protein is already oriented as desired, `orient_principal_axis=false` will suppress the automatic principal axis-based alignment.

   - Simulation system size:

```
bilayer_x_size=<default=120>
bilayer_y_size=<default=120>
```

     By default the system size is set to 120 Å along the X and Y dimensions. To change the system size, use the `bilayer_x_size` and `bilayer_y_size` parameters.

   - Position of protein with respect to the bilayer:

```
offset_z=<Protein Z-offset distance; default=0>
```

     By default, SILCS fits the bare protein into the lipid bilayer such that the center of mass of the protein is aligned with the center of mass of the bilayer. This may not be the best

choice for your membrane proteins, and you may therefore want to adjust the position of your protein according to a recommendation from an external program such as the OPM server. In this case you can adjust the relative position of the protein with respect to the position of the bilayer by using the `offset_z` option to specify the distance by which the center of mass of the protein should be offset in Z-direction from the center of the bilayer.

> **Note:** For example, to use OPM server (https://opm.phar.umich.edu/ppm_server) output as your guide to build a transmembrane system for SILCS simulation, please see *How do I fit my membrane protein in a bilayer as suggested by the OPM server?*

- Distance cutoff defining steric clashes:

```
cutoff=<cutoff for bilayer to be deleted within cutoff␣
↪distance of protein; default=3.0>
```

To avoid unfavorable steric clashes between the protein and the modeled bilayer, bilayer molecules within the distance defined by the `cutoff` parameter of the protein atoms will be deleted.

- Bilayer composition:

```
bilayer=<choice#; default=2>
```

By default, the protein will be embeded in a bilayer of 9:1 POPC/cholesterol (`bilayer=2`). Users may also choose from four other bilayer compositions using the `bilayer` option. The available bilayer compositions are pure POPC (100% POPC) (`bilayer=1`), 9:1 POPC/cholesterol (`bilayer=2`; the default), 8:2 POPC/cholesterol (`bilayer=3`), and 7:3 POPC/cholesterol (`bilayer=4`). Alternatively, users may provide their own bilayer structure using the `bilayerpdb` option.

- Custom bilayer:

```
bilayerpdb=<path to bilayer pdb file; this will override␣
↪bilayer choice>
```

The SilcsBio software provides pre-prepared bilayer membrane structures for SILCS simulations of membrane proteins. However, users may also provide their own bilayer membrane structures if desired using the `bilayerpdb` option.

The resulting protein/bilayer system will be output in a PDB file with suffix `_popc_chol`. It is important to use molecular visualization software at this stage to confirm correct orientation of the protein and size of the bilayer before using this output as the input in the next step.

Alternatively, you may use a protein/bilayer system that has been built using other software

tools.

2. **Prepare the SILCS simulation box:**

   The following command will prepare the SILCS simulation box by adding water and probe molecules to the protein/bilayer system:

```
${SILCSBIODIR}/silcs-memb/1b_setup_silcs_with_prot_bilayer prot=
↪<Protein/bilayer PDB>
```

Tip: You may use `batch=false` option to run this step on your head node. Note that it can sometimes take few hours to finish this step for very large systems.

Required parameter:

- Path and name of the input PDB file:

```
prot=<protein PDB file>
```

Optional parameters:

- Number of independent simulation systems:

```
numsys=<# of simulations; default=10>
```

- Skip the `pdb2gmx` GROMACS utility:

```
skip_pdb2gmx=<true/false; default=false>
```

- Path and name of non-covalent ligand/cofactor Mol2 files:

```
lig=<ligand MOL2 files; e.g. "lig1.mol2,/home/johndoe/
↪lig2.mol2"; default=empty/NULL>
```

  If the input structure contains ligands/cofactors and you wish to compute the SILCS FragMaps in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct position and orientation with respect to the target protein. Provide the path and name of each ligand Mol2 file separated by comma using the `lig` keyword. For covalent ligands, see additional parameters below.

Note: If your input structure contains a metal ion, please see *What happens when I set up SILCS simulations with an input structure containing a metal ion?*

- Application of weak harmonic positional restraints:

```
restraints_mode=<level of restraints on protein; flex/
↪calpha/tight/rigid/custom; default=calpha>
```

By default, a weak harmonic positional restraint, with a force constant of ~0.12 kcal/mol/Å^2, is applied to all C-alpha atoms during SILCS simulations. The `restraints_mode` keyword allows the user to customize the restraints applied to the protein.

There are four other options available:

- `flex`

  Selection: C-alpha atoms of begin and end residues of HELIX and SHEET entries in the PDB.

  Force Constant: 50 kJ/mol/nm^2 (~0.12 kcal/mol/Å^2)

  The input PDB file *must* contain HELIX and/or SHEET entries to use the `flex` restraint mode.

- `calpha`

  Selection: C-alpha atoms of all protein residues.

  Force Constant: 50 kJ/mol/nm^2

- `tight`

  Selection: Backbone atoms N, CA and C of all protein residues.

  Force Constant: 100 kJ/mol/nm^2

- `rigid`

  Selection: Non-hydrogen atoms of all residues.

  Force Constant: 150 kJ/mol/nm^2

  By default, using the `rigid` option will turn off the side chain scrambling.

  Tip: It may be beneficial to use the `rigid` option when the binding site is well-known and the protein pdb file is prepared based on a crystal structure of the ligand-bound protein and you wish to optimize the ligand with small modifications. In a retrospective study, the `rigid` option could be used when the user wants to rank a congeneric series of ligands.

- `custom`

  Selection: Custom selection of atoms provided as `restraints_string=` in the command line.

Force Constant: Custom force constant provided as `restraints_fc=` in the command line.

For example, if `restraints_string="r 17-160 | r 168-174 & a CA"` and `restraints_fc=1000`, then a force constant of 1000 kJ/mol/nm^2 will be applied to the C-alpha atoms of residues 17 to 160 and residues 168 to 174. Note that the custom selection string should follow the GROMACS selection syntax.

Warning: The `restraints_mode` keyword should be used with caution. SILCS simulations are designed to sample the local conformational space of the protein. Applying highly flexible restraints will result in poor convergence of the occupancies washing the signal from the FragMaps as we need to align the protein conformations to calculate the FragMaps. Also, large translation and rotation of the protein during MD may result in errors during trajectory collection and conversion with `gmx trjconv`.

- Reorientation of residue side chains:

  ```
  scramblesc=<true/false; default=true>
  ```

  When set `scramblesc=true`, the side chain dihedrals of the protein residues will be reoriented.

- SILCS probes with halogen-containing functional groups:

  ```
  halogen=<true/false; default=false>
  ```

  SILCS simulations can be performed with SILCS probes that contain halogen-containing functional groups by including `halogen=true`. See *Setup with halogen probes* for more details.

- Submit job to queuing system:

  ```
  batch=<true/false; default=true>
  ```

  This step is submitted to the queuing system by default. To run this step on the head node, use `batch=false`. Note that it can sometimes take few hours to finish this step for very large systems if your head node has slow CPUs and/or limited memory and/or many users are actively using the head node. It is recommended to run this step on a compute node with default `batch=true`.

Covalent Ligand Parameters:

- Add covalent ligands to the system:

  ```
  covalent=<true/false; default=false>
  ```

If the input structure contains covalent ligands and you wish to compute the SILCS FragMaps in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct state, position and orientation with respect to the target protein. See *CGenFF for Covalent Ligands Using the CLI* for more information on preparing covalent ligands for SILCS simulations.

- When covalent=true, provide the path and name of the covalent input file:

```
covalentinp=<covalent input file; default=covalent.inp>
```

Covalent input file format (5 columns separated by space):

`<lig-file-path> <linkatom1> <linkatom2> <linkresname> <linkresid>`

Where:

- `<lig-file-path>` = ligand MOL2 file path; e.g., /home/johndoe/lig2.mol2
- `<linkatom1>` = atom ID for parent atom (integer) [typically a non-hydrogen atom]
- `<linkatom2>` = atom ID for child atom (integer) [typically a hydrogen atom]
- `<linkresname>` = residue name for covalent link; CYS/LYS/ASN/SER/THR
- `<linkresid>` = residue ID for covalent link (integer)

3. **Run SILCS-Membrane GCMC/MD simulations:**

SILCS GCMC/MD simulations can now be launched with the following command:

```
${SILCSBIODIR}/silcs/2a_run_gcmd prot=<Protein/bilayer PDB> memb=true
```

Required parameter:

- Path and name of the input protein PDB file:

```
prot=<protein PDB file>
```

- Follow SILCS-simulation protocol for membrane proteins:

```
memb=true
```

The `memb` option must be set to `true` for SILCS simulations of membrane proteins.

Optional parameters:

- Simulation temperature:

```
temp=<simulation temperature; default=298>
```

The input simulation temperature should be entered in Kelvin units. By default, the temperature is set to 298 K.

- Number of cores per simulation:

```
nproc=<# of cores per simulation; default=8>
```

- Generate input files without submitting jobs to the queue:

```
batch<only generate inputs and not submit jobs true/false;
↪ default=false>
```

- Use halogen probes:

```
halogen=<use halogen probes; true/false; default=false>
```

  The SILCS simulation will be performed with halogen probes when `halogen=true`. For more information, please refer to *Setup with halogen probes*.

The SILCS GCMC/MD will begin with a 6-step pre-equilibration to slowly relax the bilayer followed by 10 ns of relaxation of the protein into the bilayer prior to the production simulation.

Tip: Membrane protein simulation systems are typically large and can require significant storage space (> 200 GB) and simulation time (> 14 days with GPUs).

---

4. **Generate FragMaps:**

   FragMap generation from the GCMC/MD simulations data follows the same protocol as for non-bilayer systems. Refer to previous instructions for `$SILCSBIODIR/silcs/2b_gen_maps` and `$SILCSBIODIR/silcs/2c_fragmap`.

If you encounter a problem, please contact support@silcsbio.com.

6.2.4 For RNA

Background

SILCS-RNA has been developed to tailor the SILCS methodology towards targeting RNA molecules with small molecules. The fundamental idea of complementarity between the functional groups in a small molecule and the binding site of the target, originally developed for protein targets, has been further developed and validated for RNA targets. Extensions to the method include an enhanced oscillating excess chemical potential protocol for the Grand Canonical Monte Carlo calculations and individual simulations of the neutral and charged solutes from which the SILCS functional group affinity maps (FragMaps) are calculated for subsequent binding site identification, pharmacophore, docking etc. calculations. Development and validation of SILCS-RNA has taken

into consideration the complexities of RNA tertiary structure and the effects of the highly negatively charged phosphate backbone on sampling of neutral and charged probes. Full details have been published in [7].

SILCS-RNA Simulations Using the SilcsBio GUI

The steps for running SILCS-RNA with SilcsBio GUI are:

1. **Begin a new SILCS-RNA project:**

   To begin a new SILCS-RNA project, select *New SILCS project* from the Home page as described in *SILCS Simulations Using the GUI* and select "RNA".

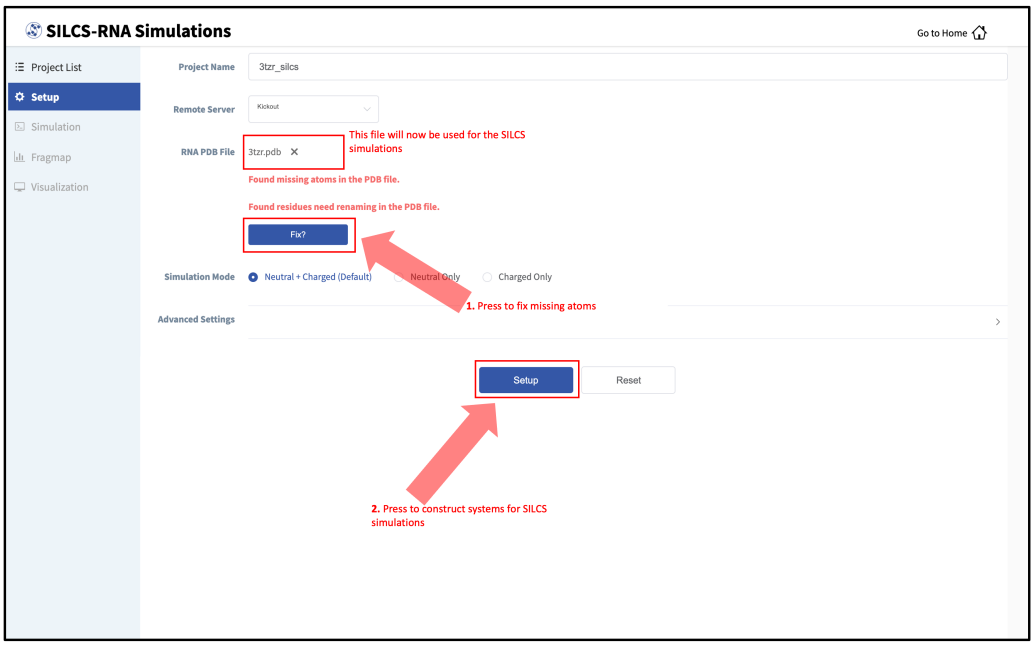
2. **Enter a project name, select the remote server, and upload input files:**

   Enter a project name and select the remote server where the SILCS-RNA simulation jobs will run. Additionally, provide the RNA input file. You may choose the RNA input file from the computer where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection*.

   If the GUI detects missing non-hydrogen atoms, non-standard residue names, or non-contiguous residue numbering, it will inform the user and provide a button labeled "Fix?". If this button is clicked a new PDB file with these problems fixed and with the suffix `_fixed` added to the base name will be created and used. The SilcsBio GUI will not model residues that are completely missing from the PDB or loops.

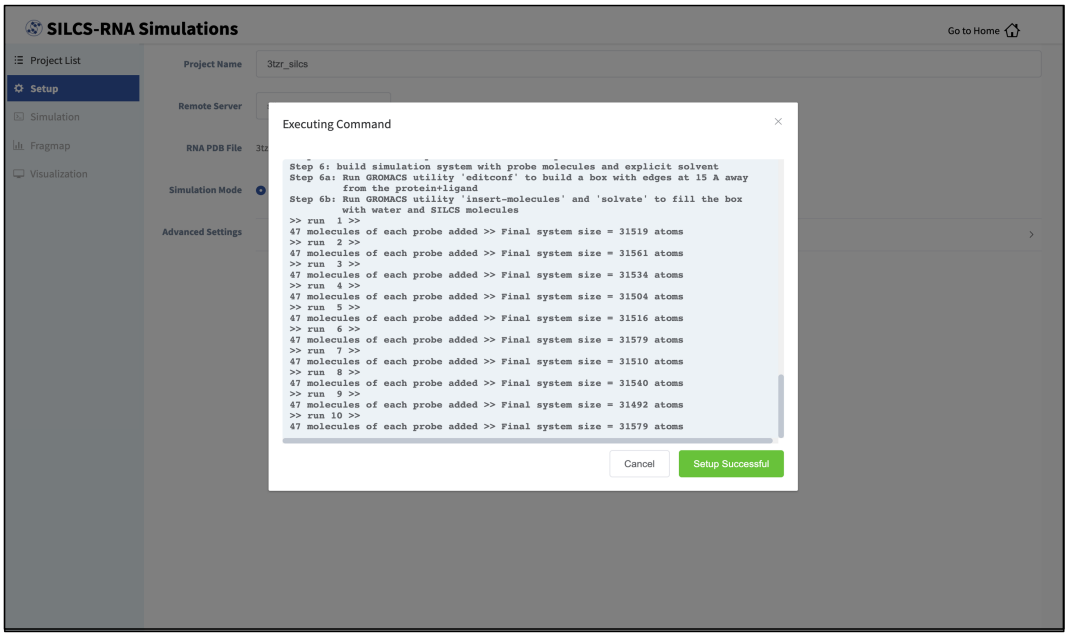
   Warning: SILCS-RNA requires a clean input RNA PDB file for the best results. The input RNA PDB file should meet the following criteria:

   - The input file should NOT include ion, solvent, or water molecules
   - The input file should only contain one MODEL. For example, PDB files from NMR studies may contain multiple models. In this case, the user must remove unwanted models and keep only one.
   - The input file should NOT contain any alternate conformations of residues.



3. **Set up SILCS simulations:**

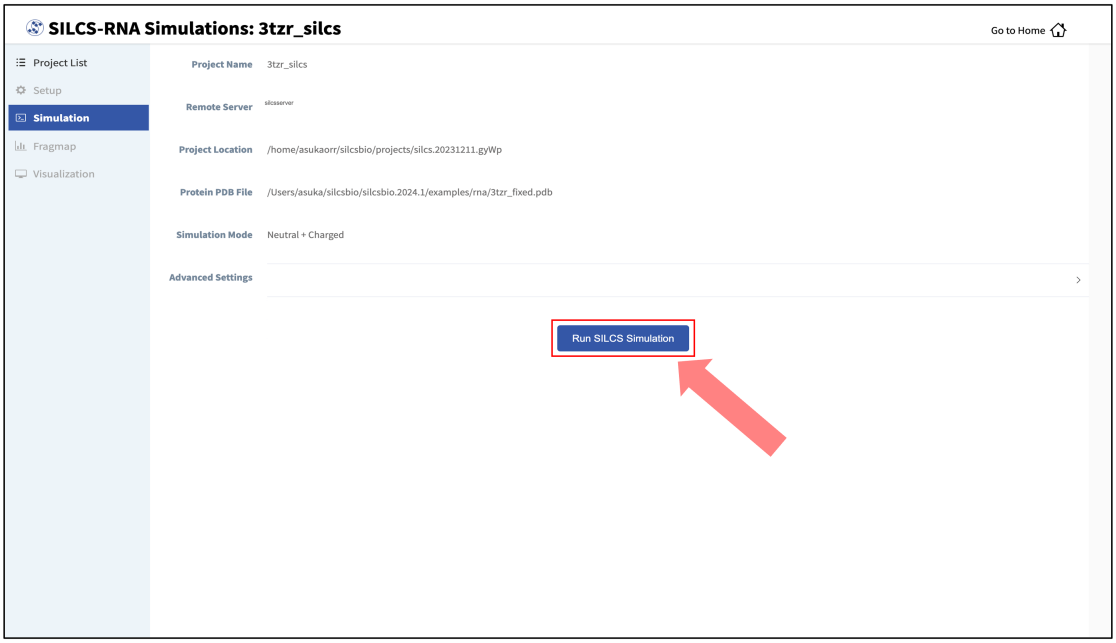
   Click the "Setup" button at the bottom of the page. The GUI will contact the remote server and perform the SILCS GCMC/MD setup process. During setup, the program automatically performs several steps: The topology of the simulation system is built and probe molecules are inserted around the RNA target. This process will produce 20 PDB files required as initial structures for the simulations and can take up to 10 minutes depending of the system size. In contrast to standard SILCS with protein targets, SILCS-RNA simulations split the GCMC/MD simulations into two separate sets, one having neutral probes and the other having charged probes. A green "Setup Successful" button will appear once the process has been successfully completed. Click the green "Setup Successful" button to proceed.

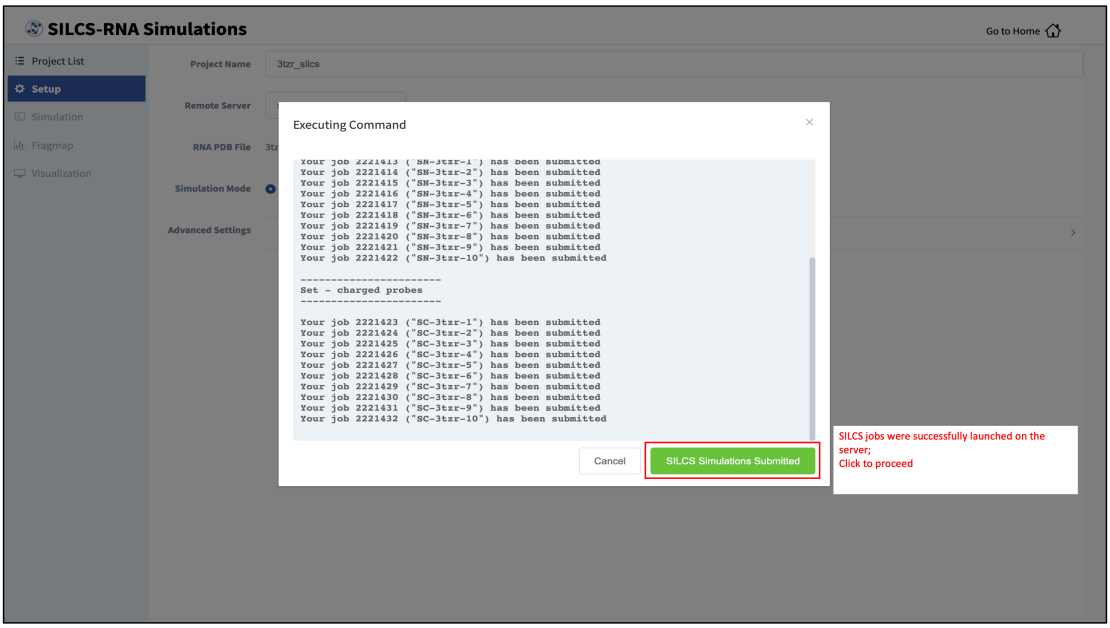


4. **Launch SILCS simulations:**

   Your SILCS-RNA GCMC/MD simulations can now be started by clicking the "Run SILCS Simulation" button. This will submit 20 compute jobs to the queueing system on the server. The GUI will indicate when the SILCS jobs are successfully launched on the server with a green "SILCS Simulations Submitted" button. At this point, the jobs will continue running on the server, and it is safe to exit the GUI.

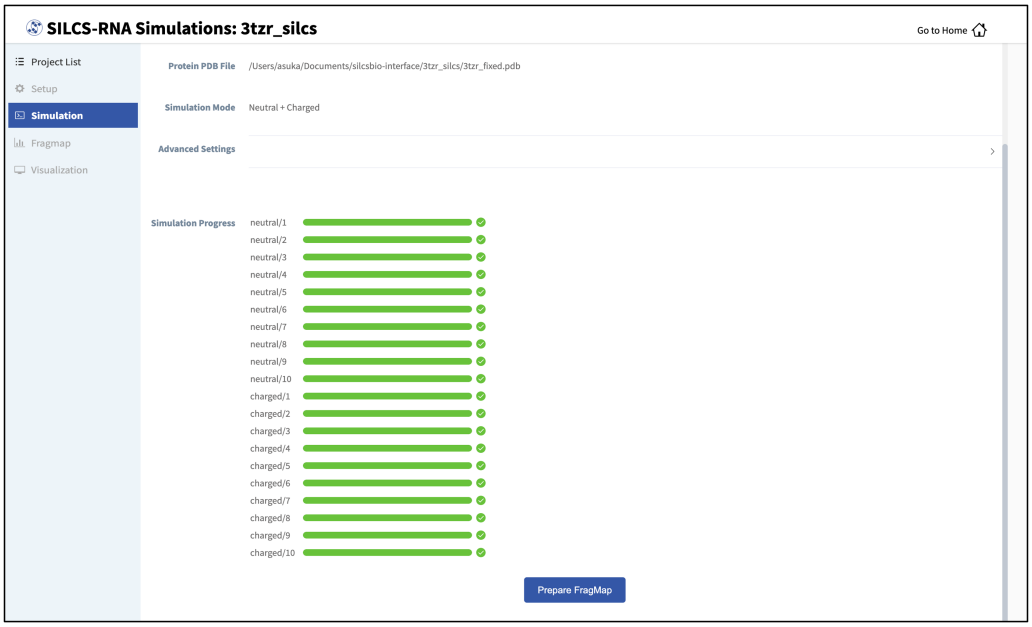
Tip: Before running your SILCS simulations, you may wish to double check that you have selected the desired file folder on the remote server and that it has 100+ GB of storage space available.



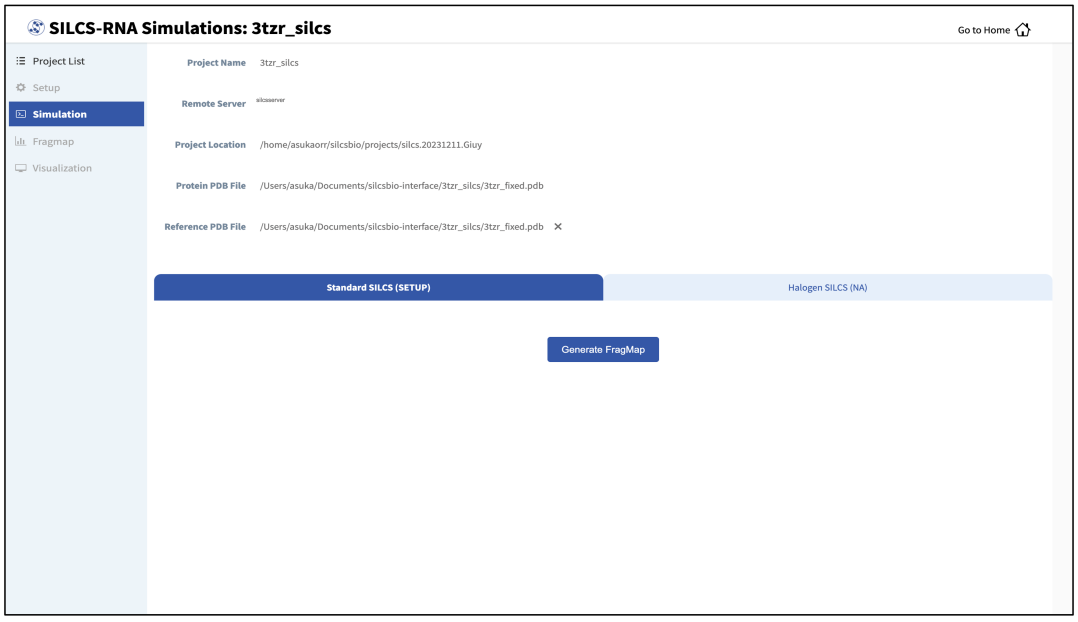


5. **Generate FragMaps:**

   Once your SILCS-RNA simulations are finished, the GUI can be used to create FragMaps and visualize them. Green progress bars indicate that the simulations have been successfully completed. Once all progress bars are green, the “Prepare FragMap” button will appear.



Press the "Prepare FragMap" button, and on the next screen, confirm your "Reference PDB File". FragMaps will be created in the coordinate reference frame of this file. Generally, this reference PDB file is the same as the initial input PDB file. A different PDB file of the same RNA structure may be used to generate FragMaps relative to that orientation. Click "Generate FragMap" to generate FragMaps and download them to the local computer for visualization.



Note: SILCS-RNA currently does not support Halogen SILCS simulations.

Processing the GCMC-MD trajectories will take 10-20 minutes, with the GUI providing updates on progress from the server during this process. Once completed, the GUI will automatically copy the files from the server onto the local computer and load them for visualization. Please see *Visualizing SILCS FragMaps* for details on how to use the SilcsBio GUI, as well as external software (MOE, PyMol, VMD), to visualize SILCS FragMaps.

As with FragMaps for protein targets, your SILCS-RNA FragMaps for RNA targets can be used for detecting hotspots and performing fragment-based drug design (*SILCS-Hotspots*), creating pharmacophore models (*SILCS-Pharm*), docking ligands and refining existing docked poses (*SILCS-MC: Docking and Pose Refinement*), and optimizing lead compounds (*SILCS-MC: Ligand Optimization*).

SILCS-RNA Simulations Using the CLI

1. **Prepare the input RNA PDB file:**

   SILCS-RNA requires a clean input RNA PDB file for best results. The input RNA PDB file should meet the following criteria:

   - The input file should NOT include ion, solvent, or water molecules
   - The input file should only contain one MODEL. For example, PDB files from NMR studies may contain multiple models. In this case, the user must remove unwanted models and keep only one.
   - The input file should NOT contain any alternate conformations of residues.

2. **Set up the SILCS-RNA simulations:**

```
${SILCSBIODIR}/silcs-rna/1_setup_silcs_boxes prot=<RNA PDB file>
```

   To determine if the setup is complete, check that the 20 PDB files required for the simulations are available using the command `ls 1_setup_*/*_silcs.*.pdb` It is also a good idea to visualize these files to confirm the presence of the RNA molecule and the SILCS solute molecules (probes) within a box of water molecules. In contrast to standard SILCS with protein targets, SILCS-RNA simulations split the GCMC/MD simulations into two separate sets, one having neutral probes (`1_setup_neutral`) and the other having charged probes (`1_setup_charged`).

   The setup command offers a number of options. The full list of parameters can be reviewed by running the command without any arguments.

   Required parameter:

   - Path and name of the input RNA PDB file:

```
prot=<RNA PDB file>
```

   Optional parameters:

- Number of independent simulation systems per set of probes:

```
numsys=<# of simulations; default=10>
```

SILCS simulations, by default, consist of 10 replicate simulations per set of probes, each with different initial velocities, to enhance sampling. SILCS-RNA simulations consist of two sets of probes, one set of uncharged probes and one set of charged probes. Thus, with both sets of probes having 10 replicate simulations, by default, 20 simulation systems will be set up (2 set of probes * 10 simulations = 20 simulations). The number of simulations can be adjusted using the `numsys` keyword, and the resulting number of simulations will be 2 * `numsys`.

- Skip the `pdb2gmx` GROMACS utility:

```
skip_pdb2gmx=<true/false; default=false>
```

- Sets of probe molecules:

```
sets=<neutral+charged/neutral/charged; default=
↪"neutral+charged">
```

Due to the highly negatively charged phosphate backbone of RNAs, SILCS-RNA uses two sets of probe molecules (one set of charged probes and one set of neutral probes) by default. The use of only one set of probe molecules may be specified with the `sets` keyword.

- Path and name of non-covalent ligand/cofactor Mol2 files:

```
lig=<ligand MOL2 files; e.g. "lig1.mol2,/home/johndoe/
↪lig2.mol2"; default=empty/NULL>
```

If the input structure contains ligands/cofactors and you wish to compute the SILCS FragMaps in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct position and orientation with respect to the target protein. Provide the path and name of each ligand Mol2 file separated by comma using the `lig` keyword.

Note: Covalent ligands are not supported in SILCS-RNA simulations. Please contact support@silcsbio.com if you wish to use covalent ligands in your SILCS-RNA simulations.

- Simulation box size:

```
margin=<system margin (Å); default=15>
```

By default, the simulation system size is determined by adding a 15 Å margin to the size of the input protein structure. The size of the simulation box can be customized by using the `margin` keyword.

Note: As in standard SILCS simulations, a weak harmonic positional restraint, with a force constant of ~0.1 kcal/mol/Å^2, is applied to all MG, C1', URA-N3, CYT-N3, ADE-N1 and GUA-N1 atoms during simulations. **The option to change this default is not available in the current version of SILCS-RNA**.

Warning: The setup program internally uses the GROMACS utility `pdb2gmx`, which may have problems processing the RNA PDB file. The most common reasons for errors at this stage involve mismatches between the expected residue name/atom names in the input PDB and those defined in the CHARMM force-field.

To fix this problem: Run the `pdb2gmx` command manually from within the `1_setup_neutral` AND the `1_setup_charged` directories for detailed error messages. For guidance on syntax, the setup script already prints out the exact commands you need to run:

```
cd 1_setup_neutral
pdb2gmx -f <RNA PDB NAME>_4pdb2gmx.pdb -ff charmm36 -water tip3p␣
↪-o test.pdb -p test.top -ter -merge all
cd ../1_setup_charged
pdb2gmx -f <RNA PDB NAME>_4pdb2gmx.pdb -ff charmm36 -water tip3p␣
↪-o test.pdb -p test.top -ter -merge all
cd ..
```

Once all errors in the input PDB are corrected, rerun the setup script with `skip_pdb2gmx=true`.

Note: SILCS-RNA currently does not support Halogen SILCS simulations.

---

3. **Submit the SILCS-RNA GCMC/MD simulations:**

   Following completion of the setup, start the 20 GCMC/MD jobs:

   ```
   ${SILCSBIODIR}/silcs-rna/2a_run_gcmd prot=<RNA PDB file>
   ```

   Required parameter:

   - Path and name of the input RNA PDB file:

```
prot=<RNA PDB file>
```

Optional parameters:

- Simulation temperature:

  ```
  temp=<simulation temperature; default=298>
  ```

  The input simulation temperature should be entered in Kelvin units. By default, the temperature is set to 298 K.

- Number of GCMC/MD cycles:

  ```
  cycles=<number of gcmc/md cycles to run; default=100>
  ```

  The number of GCMC/MD cycles can be adjusted by using the `cycles` keyword. The first cycle is number 1. The default is 100 cycles.

- Number of cores per simulation:

  ```
  nproc=<# of cores per simulation; default=8>
  ```

- Generate input files without submitting jobs to the queue:

  ```
  batch=<only generate inputs and not submit jobs true/
  ↪false; default=false>
  ```

This script will submit 20 jobs to the predefined queue. Each job runs 100 ns of GCMC/MD with the RNA and the solute molecules; 10 jobs are with neutral solute molecules (probes), and 10 are with charges solute molecules (probes).

Tip: SILCS simulations are run at a temperature of 298 K by default. It is possible to set a custom temperature by adding the `temp=<simulation temperature; default=298>` option to the `2a_run_gcmd` command. For example, `temp=310` would run the SILCS simulations at 310 K.

Once the simulations are complete, the `2a_run_gcmd_neutral/[1-10]` AND `2a_run_gcmd_charged/[1-10]` directories will contain `*.prod.100.rec.xtc` trajectory files. If these files are not generated, then your simulations are either still running or stopped due to a problem. Look in the log files within these directories to diagnose problems.

To check job progress, use the following command:

```
${SILCSBIODIR}/silcs-rna/check_progress
```

This will provide a summary list of the SILCS jobs consisting of the job path, the job number, the task ID, the job status, and the current/total number of GCMC/MD cycles. Job status values are: Q [queued], R [running], E [successfully ended], F [failed], NA [not submitted].

4. **Generate FragMaps:**

   Once your simulations are done, generate the SILCS FragMaps using the following command:

```
${SILCSBIODIR}/silcs-rna/2b_gen_maps prot=<RNA PDB file>
```

   Required parameter:

   - Path and name of the input RNA PDB file:

```
prot=<RNA PDB file>
```

   Optional parameters:

   - Simulation box size:

```
margin=<size of margin in the map; default=10>
```

   - Reference structure:

```
ref=<reference PDB; default=same PDB file given in prot>
```

     By default, the FragMaps will be oriented around the protein structure that was used as the initial structure for the SILCS simulations. The FragMaps can also be oriented around a different structure by specifying it with `ref=`.

   - Spacing between voxels:

```
spacing=<map spacing; default=1.0>
```

     The probabiliy of each functional group occupying a voxel is used to calculate GFE values. By default, the spacing between each voxel is 1.0 Å.

   - First cycle number to use for map generation:

```
begin=<cycle number to begin for map generation;␣
↪default=1>
```

   - Last cycle number to use for map generation:

```
end=<cycle number to end for map generation; default=100>
```

   This script will submit 20 jobs to the predefined queue. Each job runs 100 ns of GCMC/MD with the RNA and the solute molecules; 10 jobs are with neutral solute molecules (probes), and 10 are with charges solute molecules (probes).

This will submit 20 single-core jobs that will build occupancy maps (FragMaps) spanning the simulation box for select solute molecule atoms representing different functional groups. For a ~50K atom simulation system, this step takes approximately 10-20 minutes to complete. The FragMaps have a default grid spacing of 1 Å.

The next step is to combine the occupancy FragMaps generated from individual simulations and convert them into GFE FragMaps. Each voxel in a GFE FragMap contains a Grid Free Energy (GFE) value for the probe type that was used to create the corresponding occupancy FragMap.

```
${SILCSBIODIR}/silcs-rna/2c_fragmap prot=<RNA PDB file>
```

Required parameter:

- Path and name of the input RNA PDB file:

  ```
  prot=<RNA PDB file>
  ```

Optional parameters:

- Simulation temperature:

  ```
  temp=<simulation temperature; default=298>
  ```

  If the simulation temperature was specified for `2a_run_gcmd`, please specify the same temperature for this step using the `temp` parameter. The input simulation temperature should be entered in Kelvin units. By default, the temperature is set to 298 K.

- Skip overlap coefficient calculation:

  ```
  skipoc=<true/false; skip overlap coefficient calculation;␣
  ↪default=false>
  ```

- Generate maps in CNS format:

  ```
  cns=<true/false; generate maps in CNS format;␣
  ↪default=false>
  ```

- First cycle number to use for map generation:

  ```
  begin=<cycle number to begin for map generation;␣
  ↪default=1>
  ```

- Last cycle number to use for map generation:

  ```
  end=<cycle number to end for map generation; default=100>
  ```

GFE FragMaps will be created in the `silcs_fragmap_<RNA PDB NAME>/maps` directory, and PyMOL and VMD scripts to load these file will be created in the `silcs_fragmap_<RNA`

PDB NAME> directory. Please see *Visualizing SILCS FragMaps* for details on how to use the SilcsBio GUI, as well as external software (MOE, PyMol, VMD), to visualize SILCS FragMaps.

As with FragMaps for protein targets, your SILCS-RNA FragMaps for RNA targets can be used for detecting hotspots and performing fragment-based drug design (*SILCS-Hotspots*), creating pharmacophore models (*SILCS-Pharm*), docking ligands and refining existing docked poses (*SILCS-MC: Docking and Pose Refinement*), and optimizing lead compounds (*SILCS-MC: Ligand Optimization*).

5. **Cleanup:**

   Raw trajectory output files are large. To reduce disk usage due to these files, this step will delete unnecessary files and create a `.tgz` file for each simulation system under the `2a_run_gcmd_neutral/` and `2a_run_gcmd_charged/` directories. The command is as follows:

```
${SILCSBIODIR}/silcs-rna/2d_cleanup prot=<RNA PDB file>
```

   Required parameter:

   - Path and name of input RNA PDB file:

```
prot=<RNA PDB file>
```

   Optional parameters:

   - Number of simulation systems:

```
numsys=<# of simulations; default=10>
```

   - Delete original files:

```
delete=<delete original files after tar-balls are created␣
↪true/false; default=true>
```

     Only if the tar-balls are created successfully, the original files will be deleted when `delete=true`. Use `delete=false` to keep the original files and delete them later manually.

6.3 SILCS-Hotspots

6.3.1 Background

SILCS-Hotspots identifies all potential fragment binding sites on a target by SILCS-MC sampling of fragment-like molecules across the entire target structure [5]. The protein or other macromolecular target is partitioned into a collection of subspaces in which the fragment is randomly positioned and subjected to extensive SILCS-MC to identify favored local poses. Refer to *SILCS-MC Docking Protocol Details* for technical details on SILCS-MC sampling of fragment molecules in translational, rotational, and torsional space in the field of the FragMaps. This may be performed 1000 times or more per fragment in each subspace. The identified fragment positions are then RMSD-clustered followed by selection of the lowest energy pose in each cluster to define a binding site for further analysis.

SILCS-Hotspots may be run using small fragment-like molecules as well as larger drug-like molecules. When multiple fragments are used, a second round of clustering may be performed over the different fragment types to identify binding sites occupied by one or more of the fragment types included in the calculation.

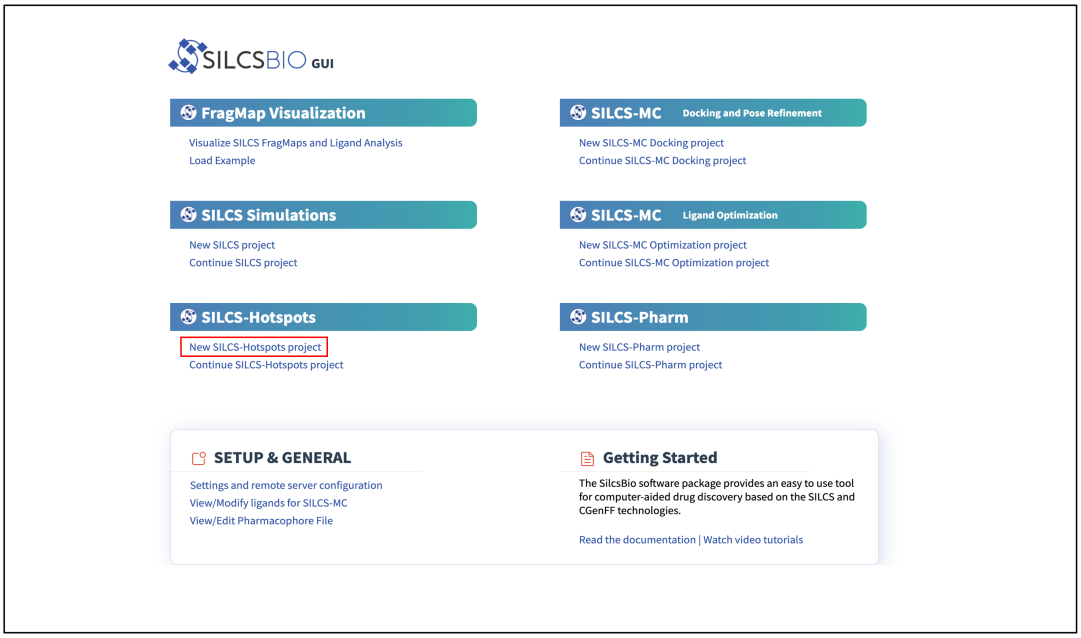
Results from SILCS-Hotspots include all potential binding poses of the individual fragments, binding sites that contain one or more of the fragment types, and ranking of fragment poses and binding sites based on LGFE scores. In addition, SILCS-Hotspots supports optional machine learning (ML)-based re-ranking of sites based on descriptors of the local binding environment. In this ML-based analysis, each site is assigned a Druggability (DB) score, defined as the decision value of the trained ML model [10]. Higher, positive DB scores indicate sites that are more likely to be druggable, while negative DB scores indicate sites that are less likely to be druggable. Applications of SILCS-Hotspots include identifying putative fragment binding sites and performing fragment-based drug design. Putative binding sites identified by SILCS-Hotspots can be used for rapid database screening with *SILCS-Pharm*. Alternatively, sites identified by SILCS-Hotspots can be considered for fragment-based design by linking poses of fragment-like molecules in adjacent sites to create drug-like molecules.

Inputs for SILCS-Hotspots are the protein or other macromolecular target used for the SILCS simulations, the SILCS FragMaps resulting from those simulations, and the fragment(s) to be used for sampling. SilcsBio provides fragment files appropriate for SILCS-Hotspots (see below). Alternatively, you may use your own fragment file(s).

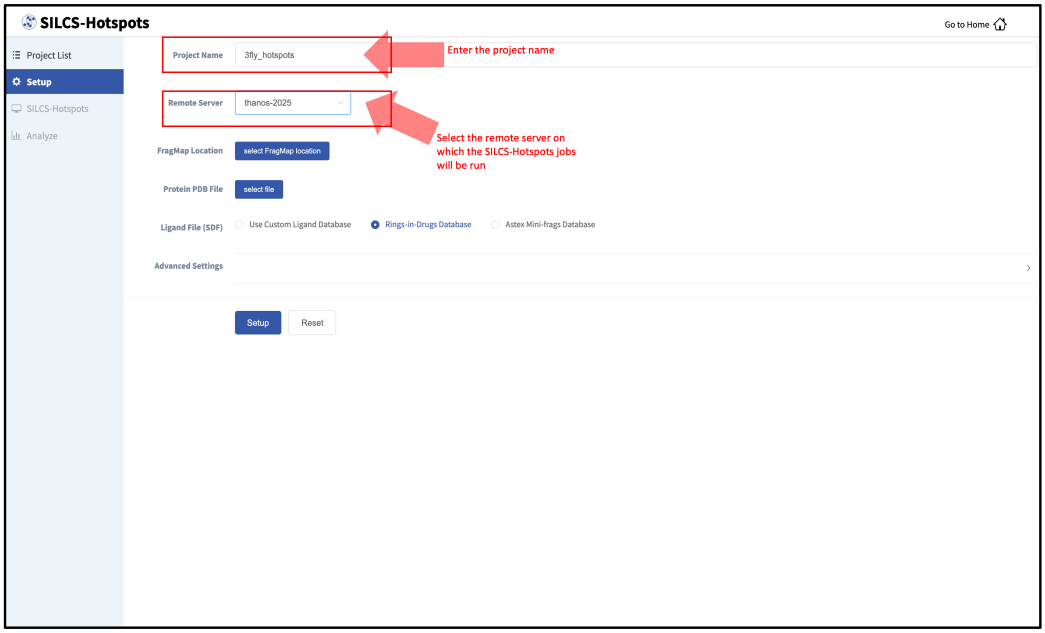
6.3.2 SILCS-Hotspots Using the SilcsBio GUI

1. **Begin a new SILCS-Hotspots project:**

   Select *New SILCS-Hotspots project* from the Home page.



2. **Enter a project name and select the remote server:**

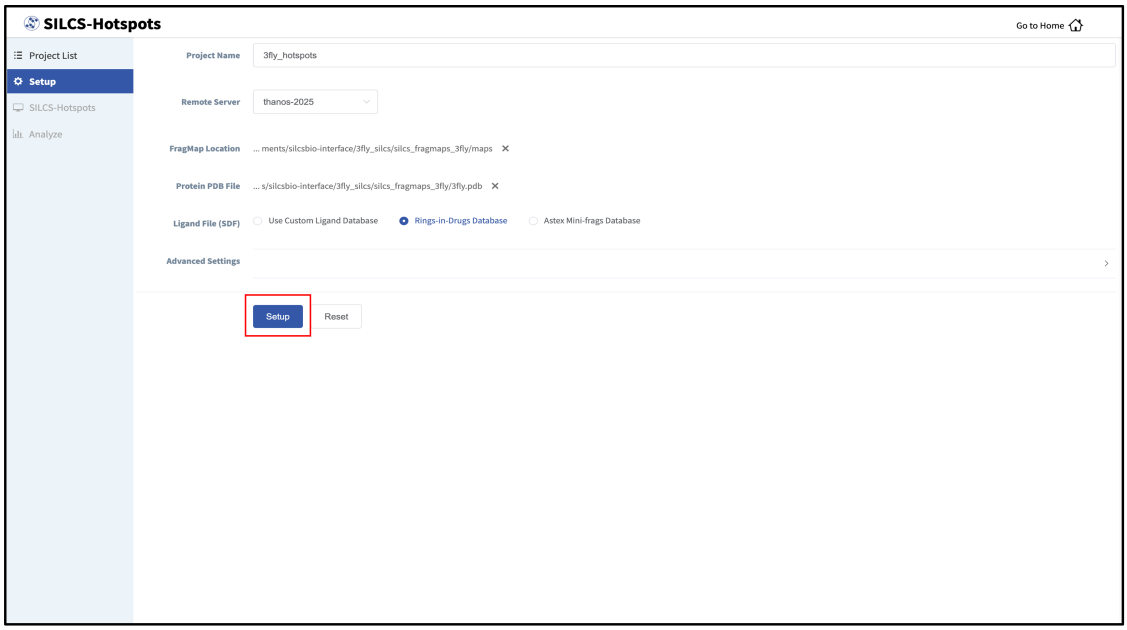


3. **Enter FragMap, protein, and ligand input files:**

   Provide FragMap and protein input PDB files. You will additionally need to provide a "Lig-

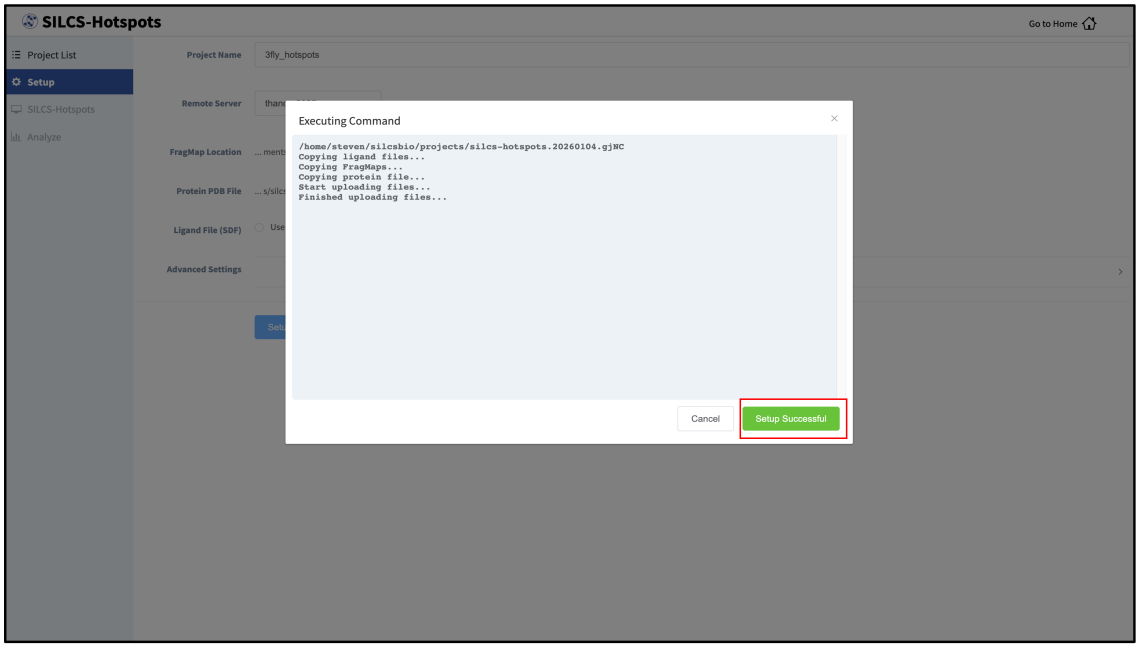
and file (in SDF format)" that contains the database of ligands to be used for sampling. You may choose these files from your local machine where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection* or you may select one of our default databases (see *Default databases of fragment-like molecules*).

> **Warning:** Ligands in the "Ligand file (in SDF format)" must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.



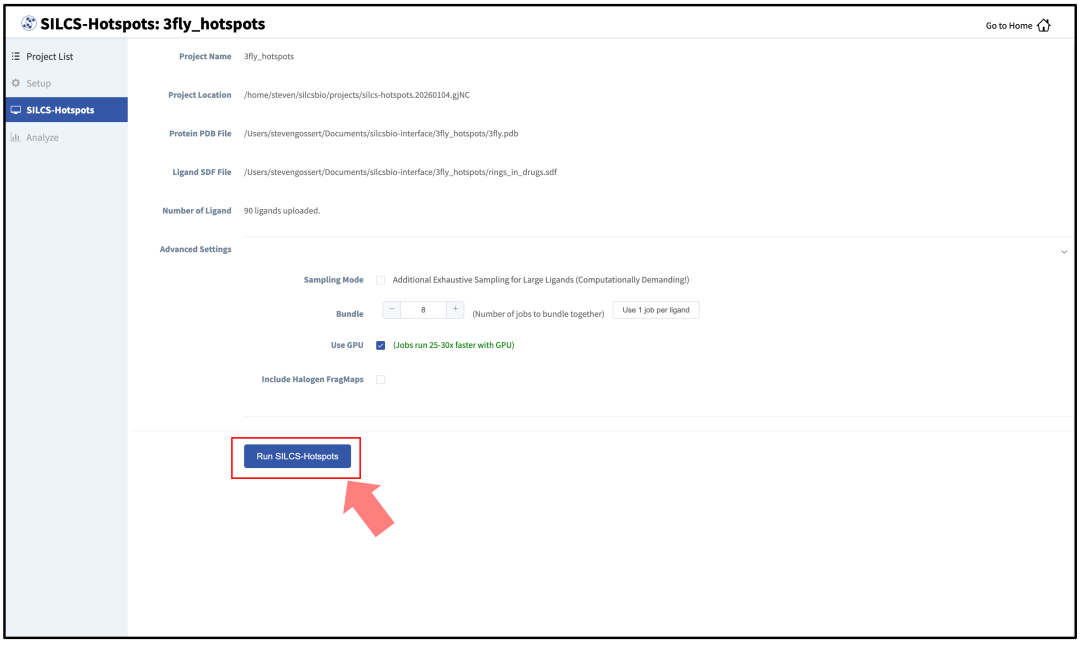
4. **Upload input files to the server:**

   Once the information is entered correctly, click the "Setup" button at the bottom of the page. The GUI will contact the remote server and upload the input files to the "Project Location" directory on the remote server. A green "Setup Successful" button will appear once the upload has successfully completed. Press this green button to proceed.

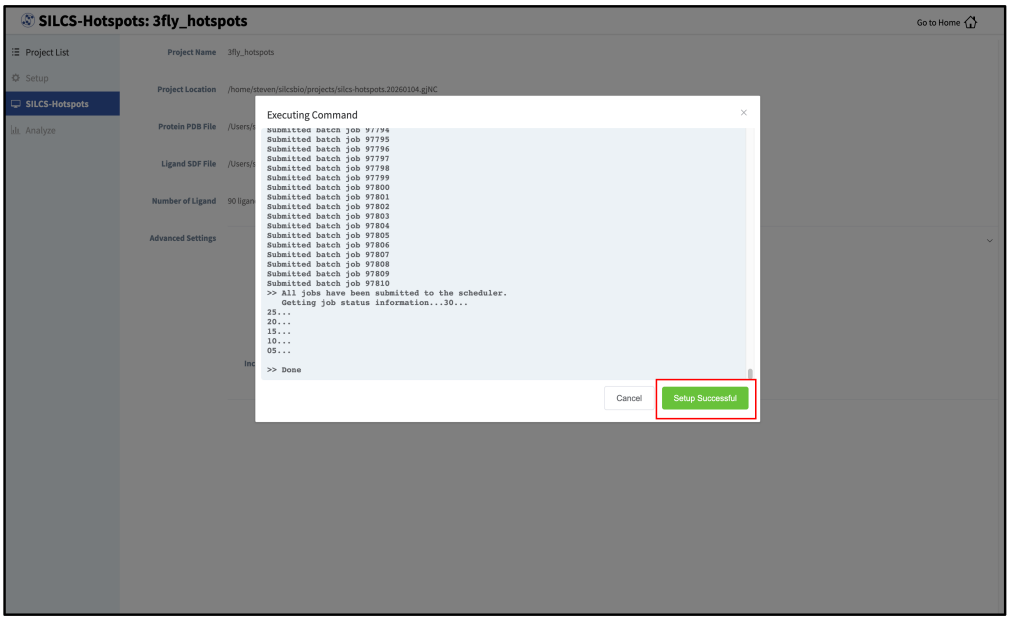


5. **Launch SILCS-Hotspots jobs:**

   The GUI will display a summary screen with the Project Name, Remote Server, Project Location, Protein PDB file, Ligand SDF file, and the Number of Ligands. You will also see "Sampling Mode", "Bundle", "Use GPU", and "Include Halogen FragMaps" under "Advanced Settings". You may select/adjust these if you desire. Click on the "Run SILCS-Hotspots" button. Doing so will submit jobs to the remote server and list them in a pop-over window.

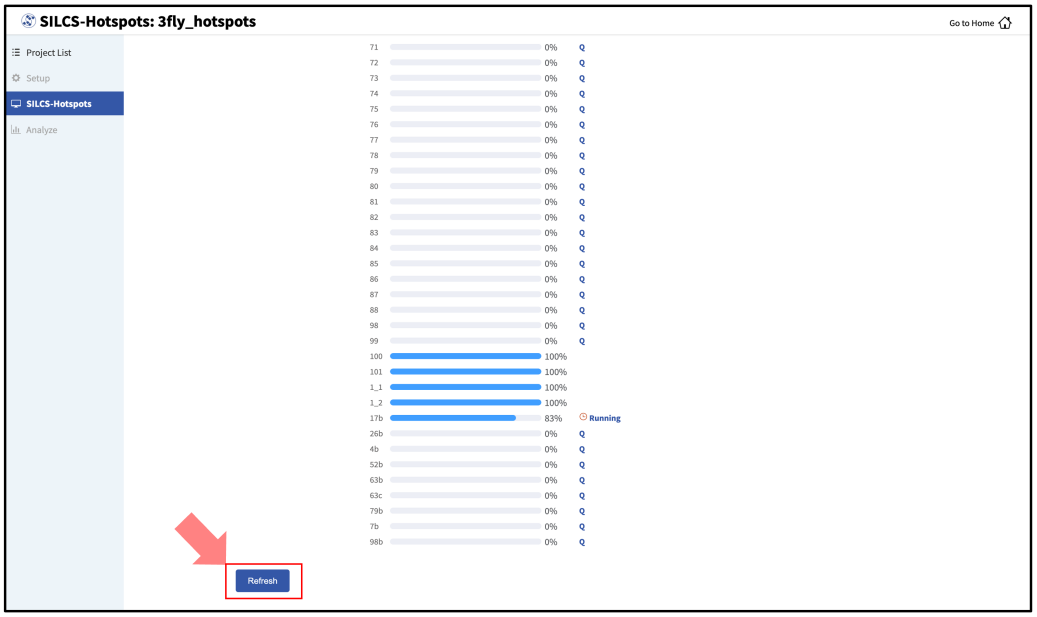


   Once the jobs are submitted, you may click on the "Setup Successful" button to dismiss the pop-over window and return to the previous screen.



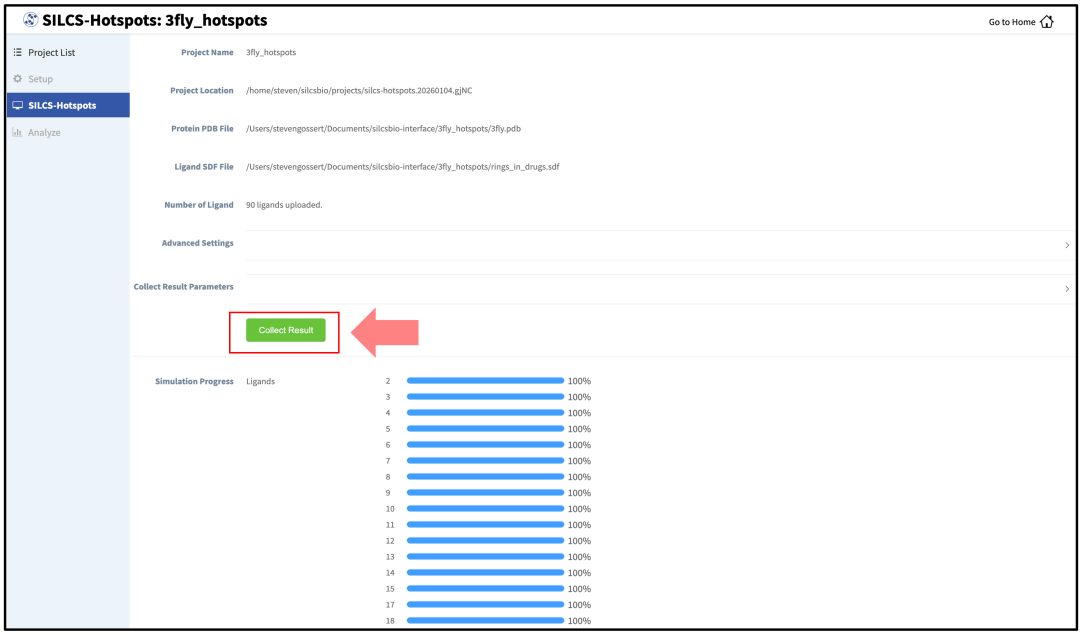
6. **Monitor SILCS-Hotspots jobs:**

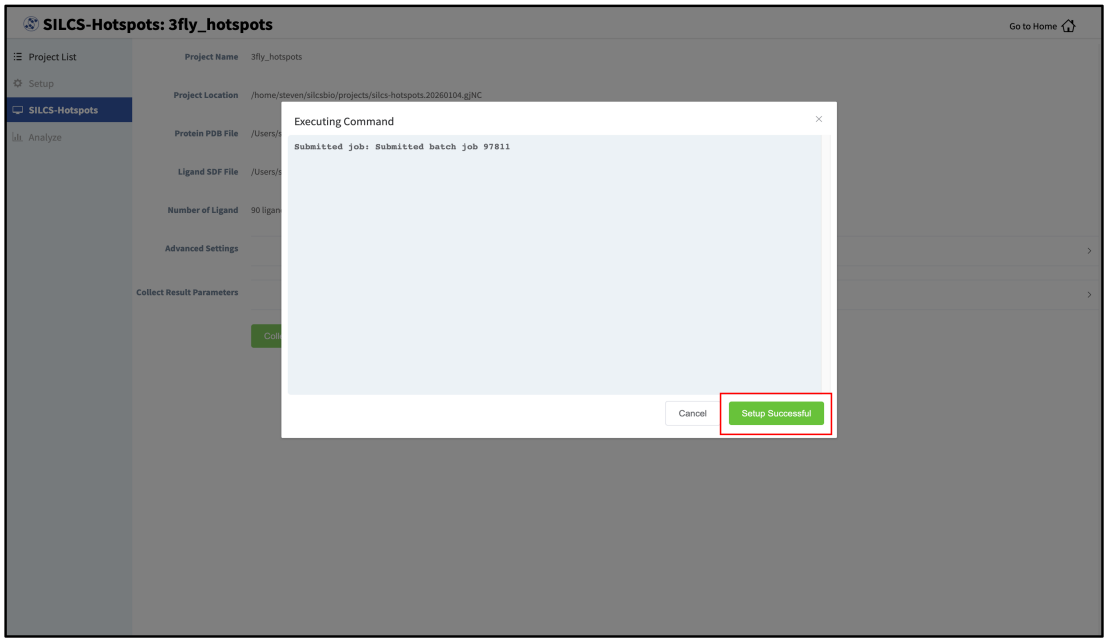
   The screen will now show a "Simulation Progress" section. You can update this section by clicking the "Refresh" button. This will update the job progress bars in the SilcsBio GUI.



7. **Collect SILCS-Hotspots data:**

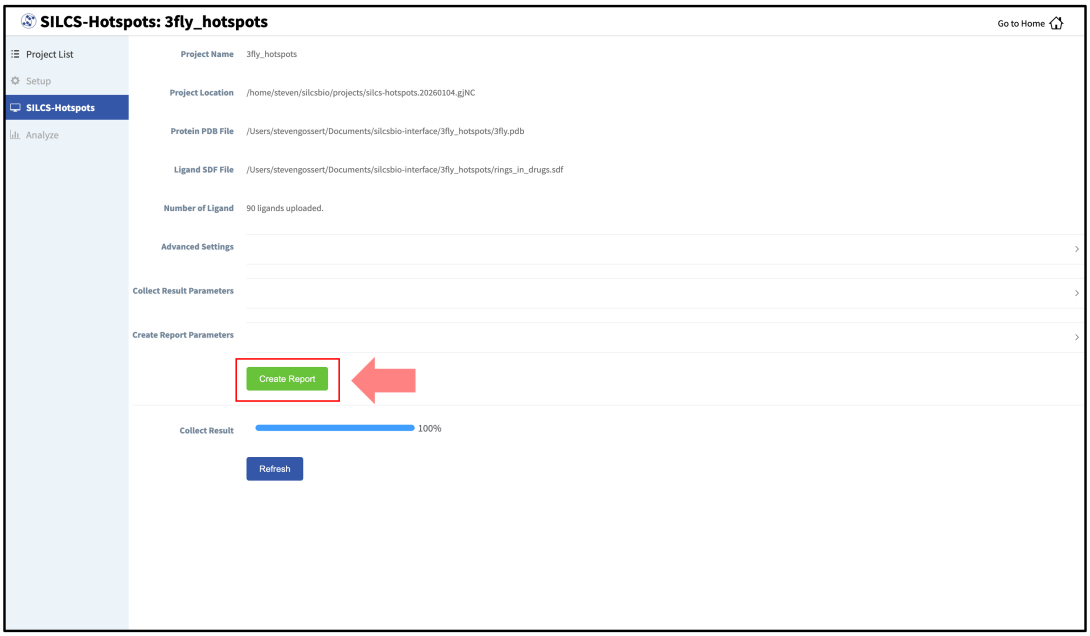
   Once the progress bars reach 100%, you will see a green "Collect Result" button. Click it to proceed. A pop-over window will appear. Once the "Setup Successful" button appears, click it to dismiss the window and return to the previous screen.

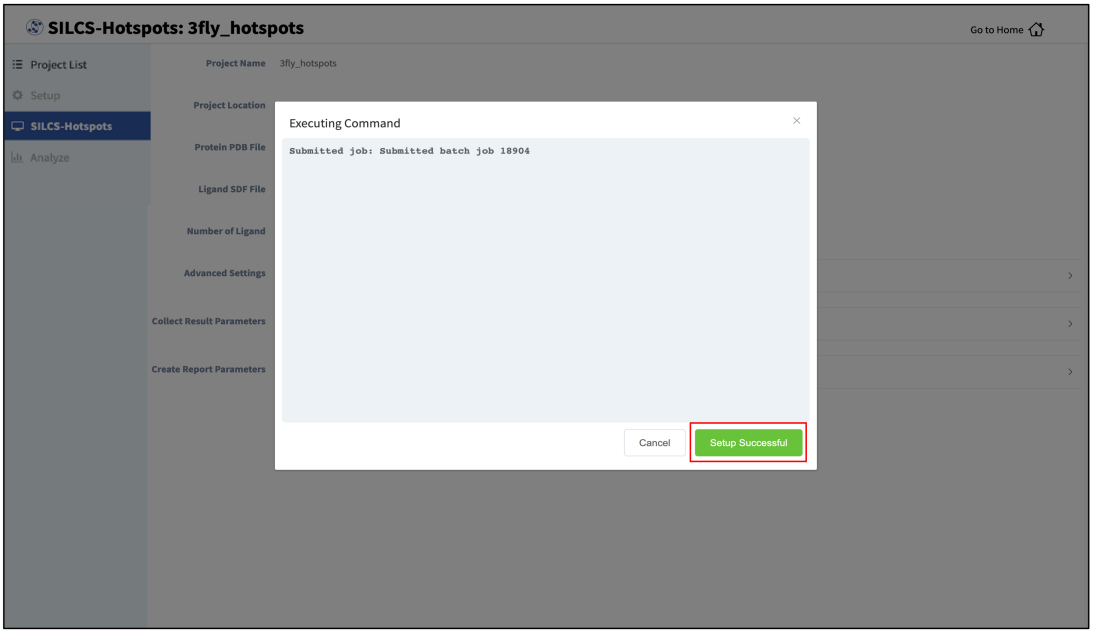




8. **Create SILCS-Hotspots report:**

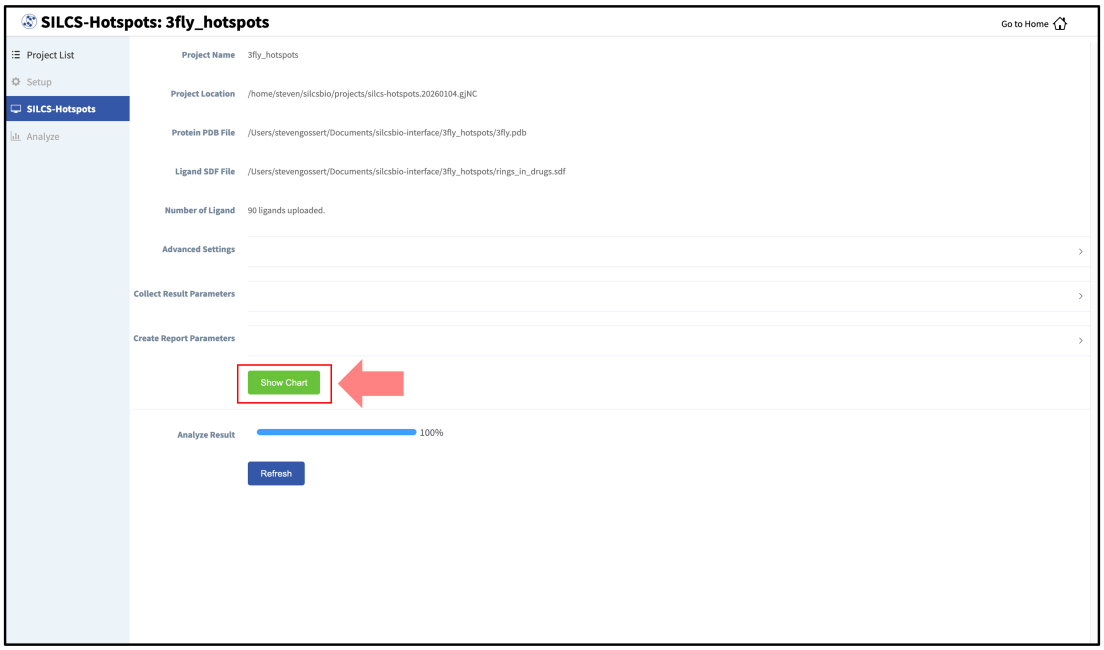
   Once the "Collect Results" progress bar reaches 100%, you will see a green "Create Report" button. Click it to submit a job to analyze the SILCS-Hotspots data and create the SILCS-Hotspots report for your project. In the pop-over window, click the "Setup Successful" button to return to the previous screen.



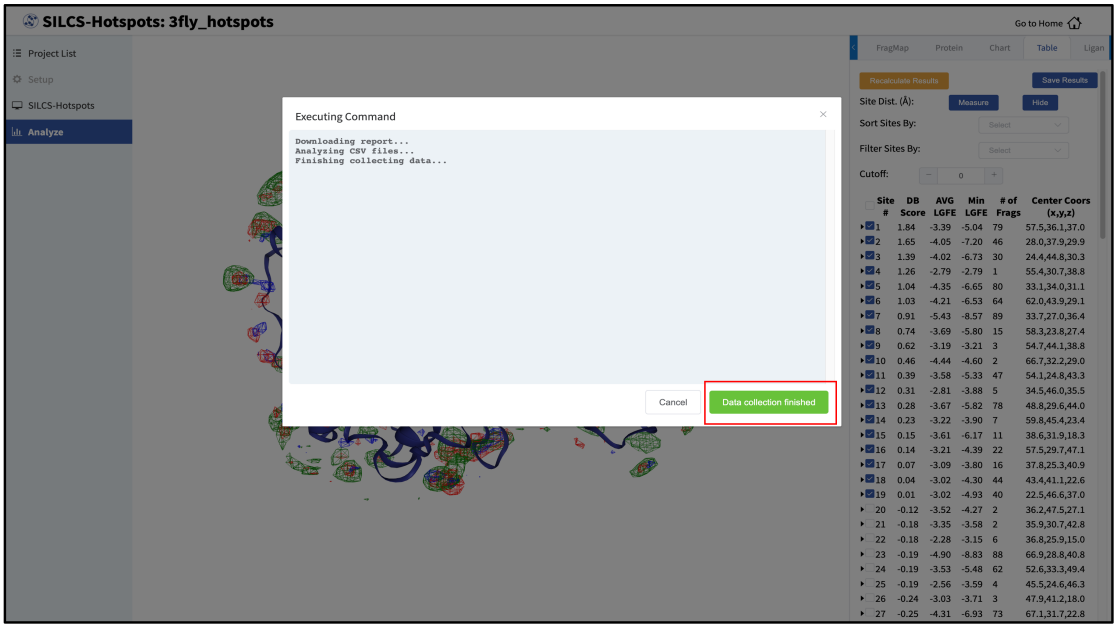


9. **Collect SILCS-Hotspots data:**

   Once the SILCS-Hotspots data is analyzed and the SILCS-Hotspots report is created, the "Analyze Result" progress bar will display 100%, and you will see a green "Show Chart" button. Click on this button to proceed.



Clicking the green "Show Chart" button will download a report folder from the server. The GUI will automatically update once all of the necessary data is collected. Click on "Data collection finished" to proceed.

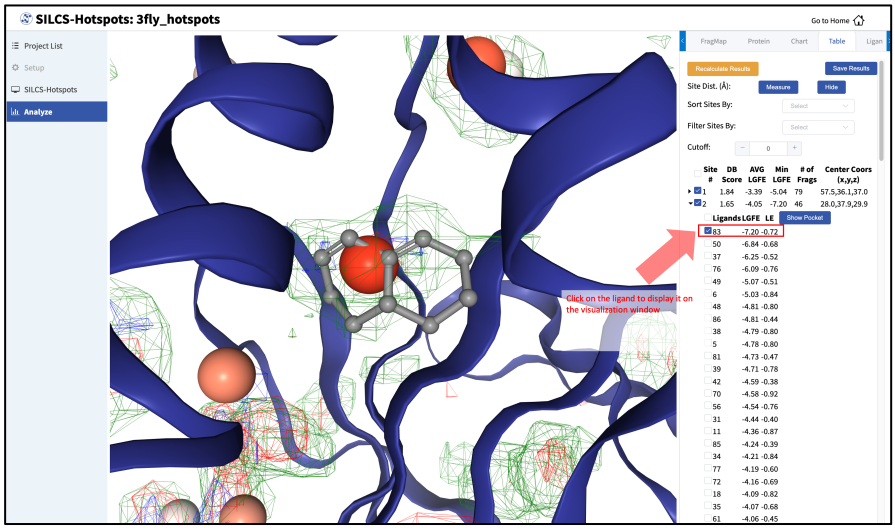


Note: If you encounter an error in this step, it is likely that required Python 3 packages have not been installed. Please refer to *Python 3 Requirement* for more information.

10. **Visualize SILCS-Hotspots results:**

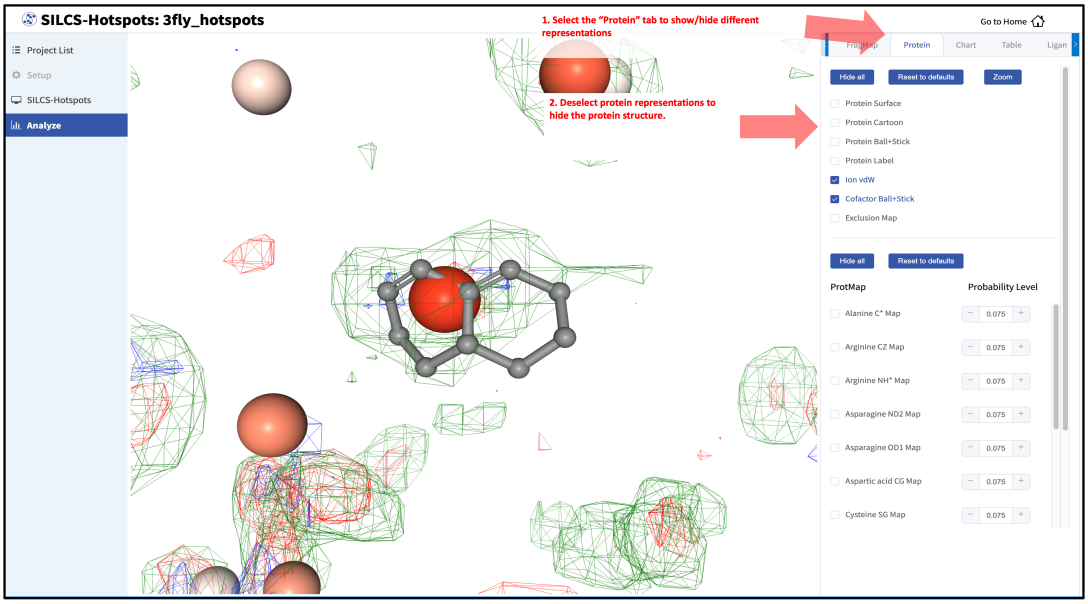
A new tab, labeled "Table" will have been created in the right-hand panel. The "Table" tab lists the Druggability (DB) Score, AVG LGFE and Min LGFE for each hotspot (referred to as "site"). Additionally, the number of fragments favorably binding to the site and the Cartesian coordinates of the site are shown.

To visualize individual fragments or ligands bound at a given site, click the arrow adjacent to the site of interest to display each individual fragment or ligand, and click on the checkbox adjacent to the desired fragment or ligand. The LGFE, and Ligand Efficiency (LE) scores for each fragment or ligand is also listed in the table. This will also center the visualization window at the ligand.



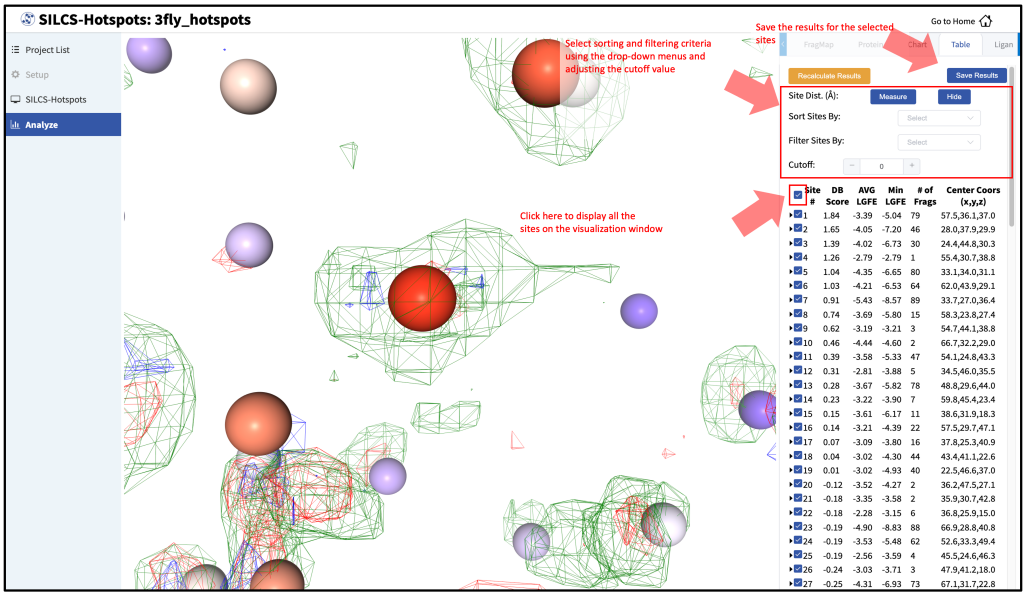
11. **Customize hidden/shown components:**

    You may also click on the "Components" tab in the right-hand panel and deselect the protein for easier viewing.



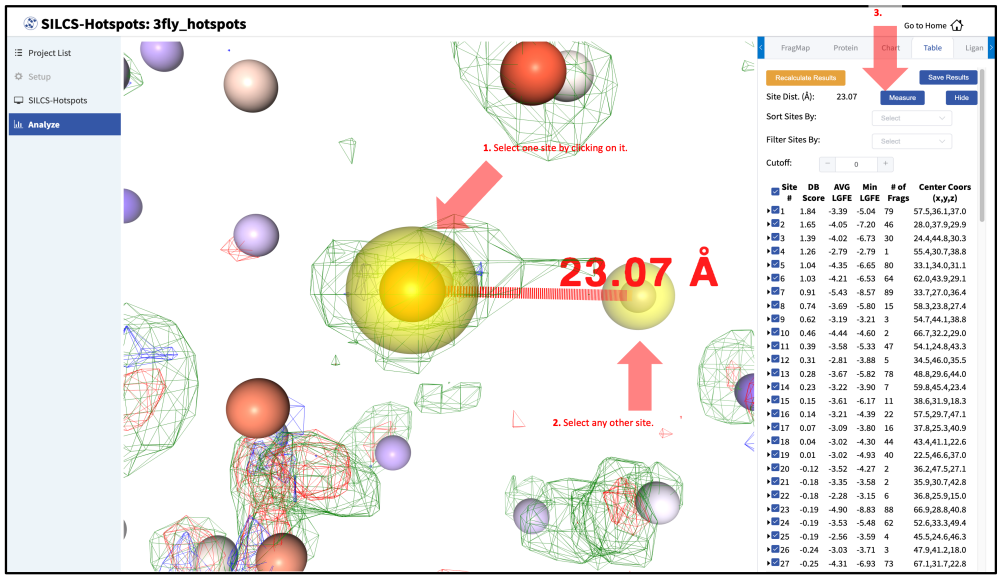
12. **Save SILCS-Hotspots docked structures and data to local computer:**

    Click on the box next to "Site #" to display all the hotspot sites in the visualization window. Click the "Save Results" button to save the SILCS-Hotspots results for all your selected hotspot sites to your local computer. You can sort the table using the "Sort Site By:" drop-down menu. You can additionally display only those sites meeting a cutoff criterion by using the "Filter Sites By:" drop-down menu and adjusting the "Cutoff:" value.



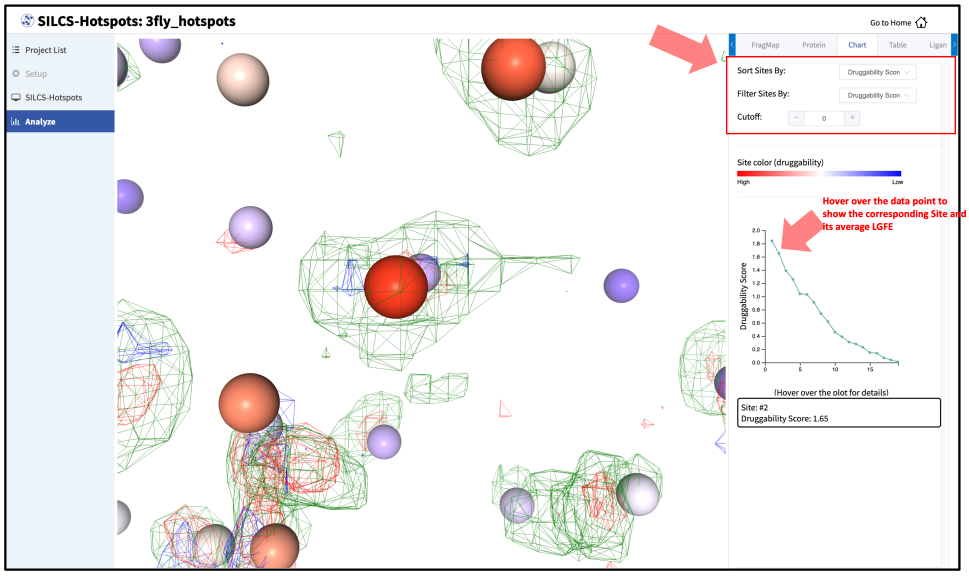
13. **Measure distances between hotspot sites:**

    To measure the distance between two hotspot sites, select their solid spheres and click the "Measure" button in the right-hand panel. This will display a dashed line in the visualization window with the measured distance (in Å) from one hotspot site to the other.



14. **Plot druggability score of SILCS-Hotspots sites:**

    The "Chart" tab in the right-hand panel converts the information from the "Table" tab into plots. The plot can be adjusted by sorting and/or filtering sites using the "Sort Sites By:" and "Filter Sites By:" drop-down menus. Changing the "Cutoff:" value will adjust the y-axis. The plot is interactive: hovering your mouse over a point will display the x and y values for that point.



6.3.3 SILCS-Hotspots Using the CLI

1. **Launch SILCS-Hotspots SILCS-MC jobs:**

    Input arguments for launching the SILCS-Hotspots SILCS-MC jobs are: the PDB file used for the SILCS run (`prot`), the location of the fragment files (`ligdir`), and the location of the SILCS FragMaps (`mapsdir`). When SILCS-Hotspots calculations are performed, the subdirectory `4_hotspots/` is created to store output from the exhaustive SILCS-MC runs.

```
$SILCSBIODIR/silcs-hotspots/1_setup_silcs_hotspots prot=<prot PDB>␣
↪ligdir=<fragment database> mapsdir=<mapsdir> bundle=<true/false>
```

Note: The above command will submit a large number of jobs, which can strain job queueing systems. The `bundle` keyword launches bundled jobs instead of individual jobs.

Required parameters:

- Path and name of the input protein PDB file:

```
prot=<protein PDB file>
```

- Fragment database location:

```
ligdir=<location and name of directory containing␣
↪fragment mol2/sdf>
```

See *Default databases of fragment-like molecules* for detailed information.

Note: `.sdf`, `.sd`, or `.mol2` files can be placed in the `ligdir` directory, and SILCS-MC will read a single molecule from each file. If a file contains multiple molecules, use of the `ligdir` option will result in only the first molecule in the file being processed.

If you have an SDF file with multiple molecules in it, replace `ligdir=<directory containing ligand mol2/sdf>` with `sdfile=<path to sdfile>` to process all molecules in the file.

Warning: Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

Optional parameters:

- FragMap directory path:

```
mapsdir=<location and name of directory containing␣
↪FragMaps; default=maps>
```

By default, the program looks for FragMaps in the `maps/` directory.

- Option to bundle jobs:

```
bundle=<true/false; default=true>
```

Note: The `1_setup_silcs_hotspots` command will submit a large number of jobs, which can strain job queueing systems. Running the command with the `bundle` keyword will submit larger bundled jobs instead of many individual jobs

---

- Number of jobs to bundle (when `bundle=true`):

```
nproc=<number of jobs to bundle; default=8>
```

- Directory containing output to be used in subsequent steps:

```
hotspotsdir=<location of hotspots data; default=4_
↪hotspots>
```

- Template file for SILCS-MC sampling:

```
paramsfile=<custom params file>
```

The default SILCS-MC job parameters file for SILCS-Hotspots is `$SILCSBIODIR/templates/silcs-hotspots/params.tmpl`. See *User-Defined Protocols through the CLI* for customization details.

- Location and name of optional SD file:

```
sdfile=<location and name of SD file>
```

Note: For SILCS-Hotspots jobs with many fragment ligands, an SD file containing all of the fragments rather than individual Mol2 files can be used.

- Exhaustiveness of SILCS-MC sampling:

```
exhaustive=<true/false; run longer sampling for larger␣
↪than fragment molecules; default=false>
```

2. **Perform post-run clustering:**

Once the SILCS-Hotspots SILCS-MC jobs are complete, the next step is to cluster fragment poses that were output by the completed jobs. The clustering algorithm iteratively finds clusters with the largest number of members [11]. The process entails 1) computing the number of neighbors of each pose, 2) choosing the pose with the largest number of neighbors and marking it and its neighbors as cluster members, 3) removing cluster members identified in Step 2 from the pool of poses, and 1) Steps 1-3 are automatically repeated until there are no available poses left. The cluster "center" is defined as that fragment pose with the most neighbors in that cluster. This pose's LGFE score defines the LGFE score of the cluster. Run the command

```
$SILCSBIODIR/silcs-hotspots/2_collect_hotspots prot=<prot PDB>␣
↪ligdir=<fragment database>
```

to perform the clustering. You may also specify the following additional options.

Optional parameters:

- Radius for RMSD clustering:

```
cutoff=<cutoff for clustering; default=3>
```

- Maximum number of sites to be identified for each fragment:

```
maxsites=<maximum number of sites ordered by LGFE;␣
↪default=200>>
```

- Directory containing output from the previous steps:

```
hotspotsdir=<location of hotspots data; default=4_
↪hotspots>
```

- Maximum allowable distance of the excipient from the protein

```
lig_max_dist=<maximum distance of ligands from the␣
↪protein; default=6>
```

- Location and name of optional SD file:

```
sdfile=<location and name of SD file>
```

Note: The SD file option should only be used if an SD file was used in the previous `1_setup_silcs_hotspots` step. In this case, use the same SD file from the previous step in the post-run clustering, `2_collect_hotspots`, step.

---

3. **Determine binding site and generate summary report:**

   The final step of SILCS-Hotspots is site determination and report generation. This involves a second round of clustering over all specified fragments to identify sites on the protein to which one or more fragments bind. Each site is given the average LGFE score over all the fragments. In addition, the sites themselves may be clustered to identify binding pockets that suggest multiple adjacent sites that may be linked to build larger fragments (under development). Information on all the fragments, the binding sites, and the binding pockets is included in `report.xlsx` and in PDB files that are output to the subdirectory `4_hotspots/report/`. Run the following command

```
$SILCSBIODIR/silcs-hotspots/3_create_report prot=<prot PDB> ligdir=
↪<fragment database>
```

   Optional parameters:

   - Directory containing output from the previous steps:

```
hotspotsdir=<location of hotspots data; default=4_
↪hotspots>
```

   - Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will␣
↪overwrite ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten.

- Path to output *report* directory:

```
reportdir=<location of report output; default=4_hotspots/
↪report>
```

- Cluster radius for binding site determination:

```
site_cutoff=<cluster radius for site determination;␣
↪default=6.0>
```

- Apply machine learning model to re-rank sites:

```
mlranking=<true/false; rank hotspots using machine␣
↪learning model; default=true>
```

- Path to python executable:

```
python=path to python executable; default=$(which python)>
```

The default python path will be checked using the `which python` command.

- Submit job to queuing system:

```
batch=<submit job to queuing system; true/false;␣
↪default=false>
```

- Number of processors to use:

```
nproc=<number of processors to use when batch=false;␣
↪default=1>
```

Additional Parameters:

- LGFE cutoff for inclusion of fragment poses in site determination:

```
ligand_lgfe_cutoff=<Ligand LGFE cutoff for site␣
↪determination; default=-2.0>
```

- Average LGFE cutoff for site determination:

```
site_lgfe_cutoff=<average LGFE cutoff for binding sites;␣
↪default=-2.0>
```

Sites having values less favorable than the cutoff will be discarded.

- Perform pocket analysis and associated clustering radius:

```
pocket=<perform pocket analysis; default=False>
pocket_cutoff=<cluster radius for binding pocket␣
↪determination; default=12.0>
```

- Option to output docked poses in PDB format:

```
pdb=<true|false; default=false>
```

When `pdb=true` the docked poses will be output in PDB format in addition to the default SD file format.

Parameters for ML ranking:

- Radius for finding adjacent hotspot clusters:

```
hscluster_radius=<radius for clustering hotspots;␣
↪default=12.0>
```

- Minimum distance for sensing exclusion map:

```
hscluster_forbidden_min=<radius for sensing exclusion map;
↪ default=1.0>
```

- Cutoffs for finding prot atoms and exclusion points for SASA calculation:

```
hsanalyze_prot_cutoff=<distance cutoff for finding prot␣
↪atoms; default=3.0>
hsanalyze_excl_cutoff=<distance cutoff for finding excl;␣
↪default=5.0>
```

- Option to calculate more information for ML ranking:

```
hsanalyze_more_info=<true/false; default=true>
```

The `3_create_report` command will populate the `4_hotspots/report/` subdirectory with the following output files:

- `report_all.xlsx`: Spreadsheet file containing analysis information. In sheets "Ligands LGFE", "Ligands LE", and "RAA" are the LGFE, LE, and relative affinity of each fragment pose obtained from clustering, respectively. In sheet "Ligands Binding Sites", all sites are listed with the average LGFE, number of bound poses, number of unique

bound ligands or fragments, minimum (most favorable) LGFE, minimum (most favorable) LE, and the LGFEs, LEs, and RAAs of each individual ligand or fragment pose in the specific site. In sheet "Ligands Binding Sites (Pivot)" each of the ligands or fragments, the site at which they bind favorably, and their corresponding LGFE at the site are listed. In sheet "Ranking", a summary of the total number of sites for each ligand or fragment is presented. If binding pocket analysis was performed, information on binding pockets is included.

- `hotspots_sites.pdb`: PDB file containing the identified sites. The B-factor column will be populated by the average LGFE score of that site.
- `pdb_by_ligands`: Directory containing SD files for each fragment with multiple coordinates for each site identified for the fragment. In the SD files, data associated with each pose will be listed, including the LGFE, LE, SMILES, atomic GFEs, and SILCS atom types. If `pdb=true` was used in `3_create_report`, then PDB files for each fragment will also be produced. In the PDB files, each pose will contain a REMARK entry including the LGFE score, and for each atom the B-factor column will include the GFE score and the final column will list the SILCS atom type.
- `pdb_by_site`: Directory containing PDB files of the fragments located at each site in `hotspots_sites.pdb`. For example, the filename `site_all_1_8_1.pdb` indicates that at site 1 fragment 8 is present. The final 1 indicates that this is the first copy of fragment 8 at site 1. From the clustering algorithm, it is possible that more than one copy of a fragment is included in a site. For example, `site_all_1_8_2.pdb` would be the second copy of fragment 8. In each PDB file, the REMARK entry includes the LGFE scores, the B-factor column includes the GFE score for each atom, and the final column lists the SILCS atom type.

The following additional outputs are produced if binding pocket analysis is performed (`pocket=true`):

- `hotspots_pockets.pdb`: PDB file containing sites that define each binding pocket. In the following, pocket P01 or 1 is defined by four sites with LGFE values for each site shown in the B-factor column. The algorithm does not number pockets consecutively: in this example, there is no P02 pocket, and pocket P03 contains 2 sites.

```
ATOM      1   X  P01 A   1      20.980   7.230 -12.513  1.00 -4.39  ␣
↪         C
ATOM      2   X  P01 A   1      27.947  16.848  -6.986  1.00 -2.82  ␣
↪         C
ATOM      3   X  P01 A   1      17.606  12.224 -10.043  1.00 -2.67  ␣
↪         C
ATOM      4   X  P01 A   1      14.912  17.557  -5.282  1.00 -2.08  ␣
↪         C
ATOM      5   X  P03 A   3      16.182 -10.455  11.793  1.00 -2.96  ␣
↪         C
ATOM      6   X  P03 A   3      17.605   0.058  15.384  1.00 -2.83  ␣
```

(continues on next page)

(continued from previous page)

```
↪          C
```

- `pdb_by_pocket`: Directory containing PDB files for fragments that comprise each pocket. For example, `pocket_all_8_site_12_10_1.pdb` indicates pocket 8 contains a fragment from site 12 and that fragment is fragment 10. 1 indicates that it is the first copy of fragment 10 in that pocket. In each PDB file, the REMARK entry includes the LGFE scores, the B-factor column includes the GFE score for each atom, and the final column lists the SILCS atom type.

The following additional outputs are produced if Machine Learning ranking analysis is performed (`mlranking=true`):

- `hotspots_sites_mlranking.pdb`: PDB file containing the identified sites ranked by the machine learning model [10]. The B-factor column lists the decision value of the machine learning model (the druggability score). The occupancy column value includes the original ranking (based on LGFE) of the site. Higher, positive druggability scores indicate that a site is druggable. Negative druggability scores indicate that a site is less likely to be druggable.

Note: If you encounter an error in steps 2 or 3, it is likely that required Python 3 packages have not been installed. Please refer to *Python 3 Requirement* for more information.

6.3.4 Retraining the ML Model

The ML model as built into the SILCS-Hotspots workflow was trained on the rings-in-drugs database, if you are using a different database you may want to retrain your ML model. The script for training a new model is located in the following directory:

```
${SILCSBIODIR}/utils/python/silcs-hotspots/ml_ranking/
```

And after generating your training data and making sure it is in the data_hstrain directory, the script can be run using the following command:

```
python ${SILCSBIODIR}/utils/python/silcs-hotspots/ml_ranking/hstrain.py
```

This will generate the pkl files for the trained model in the same directory so it is recommended that users move the original pkl files (model.pkl and pca.pkl) to a separate directory to avoid overwriting the original models. This is summarized in a readme file in the same directory.

6.3.5 Practical considerations

`1_setup_silcs_hotspots` creates a subdirectory, `4_hotspots/`, (or user defined name using `hotspotsdir=`) that contains fragment pose and LGFE information. As SILCS-Hotspots makes use of large numbers of SILCS-MC calculations on each fragment, this directory will be filled with a substantial amount of data. It is suggested that, once all analyses are complete, these files either be deleted or archived, and `4_hotspots/` be renamed prior to additional SILCS-Hotspots runs. Remember, the data in `4_hotspots/` are used for post-run clustering and site determination and report generation.

If `1_setup_silcs_hotspots` is being rerun with new parameters, such as new fragments as specified by `ligdir=`, the subdirectory `4_hotspots/` should be renamed or an alternate name assigned using `hotspotsdir=` to avoid information from the original run being overwritten. However, this is not strictly necessary IF all the new fragments have unique filenames relative to the original run AND the SILCS-MC job parameters specified by `paramsfile=` are not changed, since the subsequent `2_collect_hotspots` command only performs analysis on Mol2/SDF files in the specified `ligdir` directory.

`1_setup_silcs_hotspots` launches a large number of jobs; while the majority of these jobs finish quickly, individual jobs may take time (minutes to an hour) to complete. In some cases one or two jobs in the set may require additional time due to the robust SILCS-MC convergence criteria used in SILCS-Hotspots.

For the SILCS-MC sampling, especially with larger fragments or ligands, you may prefer not to include the SILCS Exclusion Map. This will permit fragment sampling of poses that would otherwise be rejected because of overlap with the exclusion region. Using this approach, final fragment poses will be based solely on scoring with SILCS FragMaps. To achieve this, set the weighting of the Exclusion Map to zero in your custom SILCS-MC job parameters file (`paramsfile=<custom params file>`). A simple means to this end is to use a copy of the default file `$SILCSBIODIR/silcs-mc/params_custom.tmpl` in which you replace the `1.000` in the below line with `0.000`.

```
SILCSMAP EXCL <MAPDIR>/<prot>.excl.map        1.000
```

Clustering of data in the `4_hotspots/` or equivalent subdirectory with the `2_collect_hotspots` command produces representative fragment poses on a per-cluster basis. If clustering is repeated, for example with a different clustering radius, the old PDB files created in `4_hotspots/` will be overwritten. Therefore, make sure to run site determination and report generation (the `3_create_report` command) on your original clustering output before repeating clustering.

`2_collect_hotspots` processes each of the fragments individually and typically requires minutes to complete for 100 fragments. During that process a number of warning messages, such as "Clustered PDB not found: 4_hotspots/2/subspace_1/pdb/2_clust_1_1.pdb," will be given. These are expected as they indicate subspaces for the fragments in which no favorable fragments poses were identified. For example, this may occur where the subspace encompasses the protein structure.

Site determination and report generation with the `3_create_report` command creates a subdirectory, `4_hotspots/report/`, and outputs `report.xlsx` and PDB files into this subdirectory.

Rerunning `3_create_report` with different parameters, like an alternate value of the site clustering cutoff, will overwrite the original output. Therefore, prior to rerunning report generation, rename `4_hotspots/report/` to save your original report generation outputs. The clustering algorithm for site determination does not consider the identity of the fragment when performing the clustering. Accordingly, in certain cases it is possible for the same fragment to be included two or more times in a given site.

6.3.6 Example

The following demonstrates use of SILCS-Hotspots on p38 MAP kinase. Input files, including FragMaps, are in `$SILCSBIODIR/examples/silcs/`.

```
cp -r $SILCSBIODIR/examples/silcs/silcs_fragmaps_3fly .
cd silcs_fragmaps_3fly
$SILCSBIODIR/silcs-hotspots/1_setup_silcs_hotspots prot=3fly.pdb ligdir=
↪$SILCSBIODIR/data/databases/ring_subset mapsdir=maps bundle=true
```

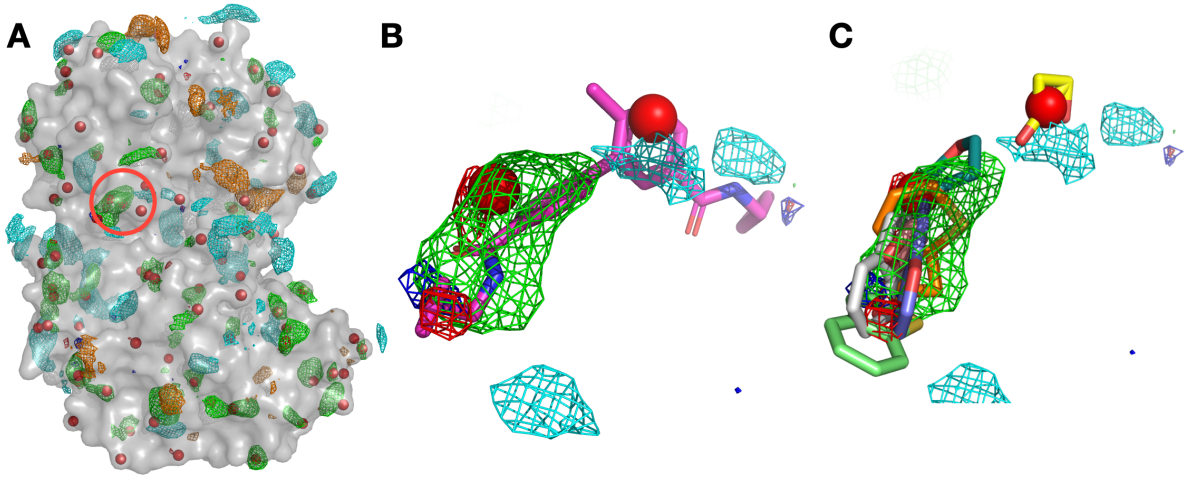
In this example, we use `$SILCSBIODIR/data/databases/ring_subset` as the fragment database (see *Default databases of fragment-like molecules*). Owing to the small size of this database, the example run finishes quickly.

Once the jobs spawned by `1_setup_silcs_hotspots` complete, use the following commands for post-run clustering and site determination and report generation.

```
$SILCSBIODIR/silcs-hotspots/2_collect_hotspots prot=3fly.pdb ligdir=
↪$SILCSBIODIR/data/databases/ring_subset
$SILCSBIODIR/silcs-hotspots/3_create_report ligdir=$SILCSBIODIR/data/
↪databases/ring_subset
```

This will have created the `4_hotspots/report/` subdirectory containing the following:

- `hotspots_sites.pdb`: Centroid positions of fragment clusters, that is, the "hotspots." Clusters are ranked using average LGFE scores, with the cluster rank listed in the residue number field and the average LGFE score in the B-factor field.
- `report_all.xlsx`: Report of hotspots in spreadsheet format.
- `pdbs_by_sites`: List of PDB files for each hotspot.



Panel (A) shows p38 (surface representation), SILCS FragMaps (wire frame), and all hotspots (red spheres) determined by SILCS-Hotspots. The crystal binding pocket is circled in red. Panel (B) has FragMaps, hotspots, and the crystallographic ligand. Two hotspots are identified within the crystal binding pocket and coincide with the rings of the ligand. Panel (C) shows fragment poses from SILCS-Hotspots calculations as well as FragMaps and hotspots.

6.3.7 Validating hotspots using FDA-approved drugs

A SILCS-Hotspots run will typically use a database of fragment-like molecules (see *Default databases of fragment-like molecules*) to determine probable small-molecule binding hotspots on the input target. Further validation of a subset of these hotspots can be done by docking and scoring actual FDA-approved drugs at these hotspot locations. To this end, an automated workflow is provided for docking 348 diverse FDA-approved drugs at user-selected hotspots and calculating the drug molecules' relative buried surface areas (rBSA) before and after docking.

Note: The percent relative buried surface area (%rBSA) of a ligand is defined as

$$\%\text{rBSA} = 100 - 100 \times \frac{\text{SASA}_{\text{bound}}}{\text{SASA}_{\text{unbound}}}$$

where $\text{SASA}_{\text{unbound}}$ is the surface accessible surface area of the unbound ligand and $\text{SASA}_{\text{bound}}$ is the surface accessible surface area of the bound ligand.

A high %rBSA of a ligand indicates that the ligand is well encased in the protein binding pocket. A low %rBSA of a ligand indicates that the ligand is bound on the surface of the protein. In general, a high %rBSA is desirable.

Suggested characteristics of hotspots to be selected for further validation

Users will need to judiciously choose a subset of the hotspots determined by SILCS-Hotspots for further validation by docking of FDA-approved drugs. As these drug molecules are significantly larger and have more rotatable bonds than the fragment-like molecules used in determining hotspots, there is relatively more computational expense associated with docking them versus fragment-like molecules. This can make it impractical to apply the validation procedure to the entire set of hotspots. Suggested characteristics for choosing a hotspot to include in your subset for further validation include

- The hotspot is partially or fully buried
- The hotspot has other hotspots nearby (typically within 10 Å)
- The hotspot has 2 or more apolar FragMap regions in close vicinity
- The hotspot has multiple hydrogen bond Don/Acc FragMap regions in close vicinity

The following steps describe the SILCS-Hotspots validation procedure.

1. **Set up and run SILCS-MC simulations at selected SILCS-Hotspots:**

   Once you have selected your subset of hotspots, you will need to EITHER supply them as a separate pdb file <my_hotspots.pdb> OR as a comma-separated list (e.g. "2,7,10,22") along with the full hotspots_sites.pdb file from the SILCS-Hotspots standard report:

   ```
   $SILCSBIODIR/silcs-hotspots/4a_fda_top20_analysis prot=<prot PDB>␣
   ↪hotspotspdb=<my_hotspots.pdb>
   ```

   OR

   ```
   $SILCSBIODIR/silcs-hotspots/4a_fda_top20_analysis prot=<prot PDB>␣
   ↪hotspotspdb=hotspots_sites.pdb myhotspots="2,7,10,22"
   ```

   This first step will set up and run SILCS-MC simulations of each molecule in a database of 348 FDA-approved drug molecules for each selected hotspot. The SILCS-MC in this step entails exhaustive sampling of ligand orientations and conformations.

   Optional parameters:

   - A tag identifying the subset of hotspots you are analyzing:

     ```
     sitetype=<tag_name>
     ```

     The default <tag_name> is set to "rings". If you use this option, make sure to use it for ALL the following steps as well.

   - FragMap directory path:

```
mapsdir=<location and name of directory containing␣
↪FragMaps; default=maps>
```

By default, the program looks for FragMaps in the `maps/` directory.

- Compound library/database location:

```
sdfile=<location and name of SDF containing all compounds;
↪ default=$SILCSBIODIR/data/databases/fda_approved_348.
↪sdf>
```

or

```
ligdir=<location and name of directory containing␣
↪compound mol2/sdf>
```

The default database is a database of 348 diverse FDA-approved drug molecules, which is expected to capture the chemical space of existing drug molecules. However, you may choose to supply a different database.

- Option to bundle jobs:

```
bundle=<true|false; default=true>
```

Multiple (single) jobs will be bundled into a single larger job by default (`bundle=true`). The number of jobs to be bundled can be set with the `nproc` parameter.

- Number of jobs to bundle:

```
nproc=<number of jobs to bundle; default=8>
```

2. **Collect SILCS-MC results:**

The second step involves collecting poses from the completed SILCS-MC simulations of the 348 FDA-approved drug molecules and rank ordering them. The collected results are output to `lgfe.csv`.

```
$SILCSBIODIR/silcs-hotspots/4b_fda_top20_analysis prot=<prot PDB>␣
↪hotspotspdb=<my_hotspots.pdb>
```

3. **Compute rBSA:**

The third step will set up and submit jobs to compute rBSA values for top compounds.

```
$SILCSBIODIR/silcs-hotspots/4c_fda_top20_analysis prot=<prot PDB>␣
↪hotspotspdb=<my_hotspots.pdb>
```

Optional parameter:

- Number of top compounds for subsequent analysis

```
top=<number of top results for analysis; default=20>
```

The compounds are rank-ordered by their LGFE score in ascending order (most favorable to least favorable), and the top compounds are selected for subsequent analysis. By default, this is set to `top=20` as suggested by the names of the job scripts in this section. Thus, the top 20 compounds will be selected by default. If you change this default, make sure to include the same option in the next final step.

4. **Collect final rBSA and LGFE results:**

The fourth and final step will collect rBSA results, plot rBSA-vs-LGFE, and create a directory `fda_top20_analysis` containing the final results.

```
$SILCSBIODIR/silcs-hotspots/4d_fda_top20_analysis prot=<prot PDB>␣
↪hotspotspdb=<my_hotspots.pdb>
```

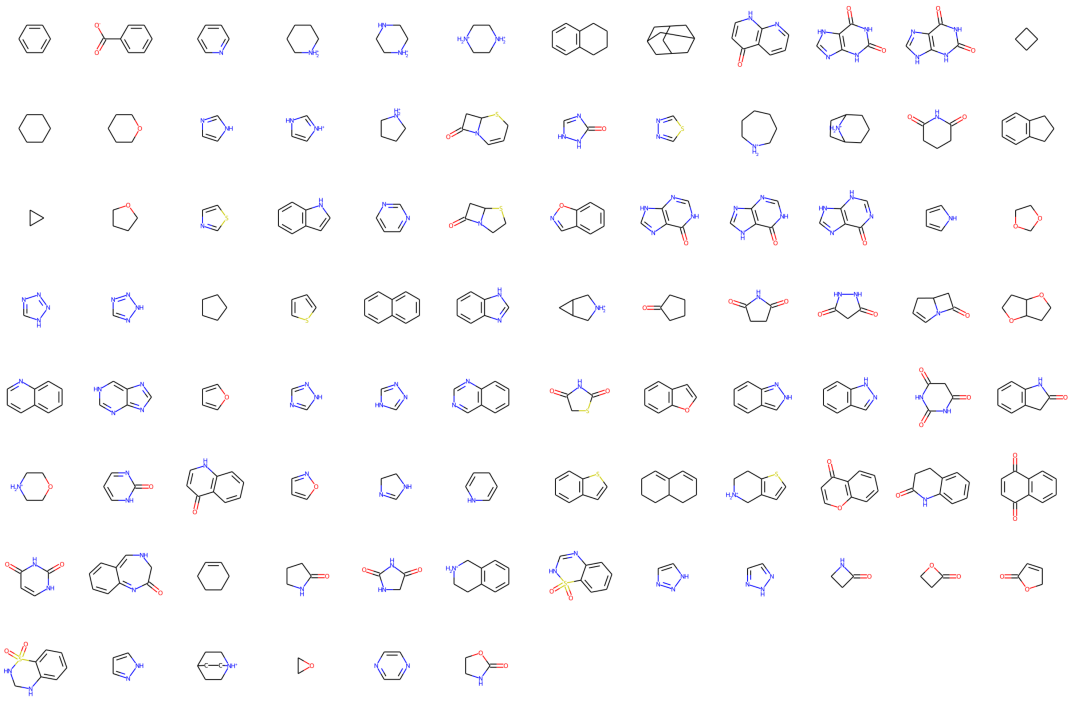
To evaluate a different subset of hotspots, run the above steps from the beginning with the new subset of hotspots and a different `sitetype` value specified in the first step, *Set up and run SILCS-MC simulations at selected SILCS-Hotspots*.

> **Warning:** Re-using the same `sitetype` value will overwrite any previous results.

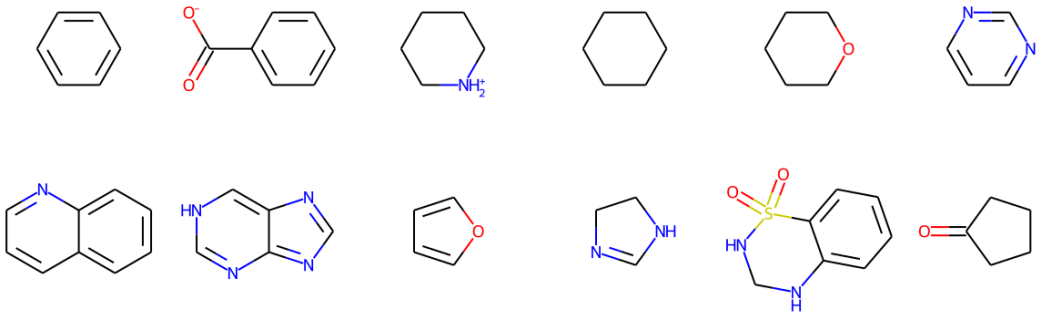
6.3.8 Default databases of fragment-like molecules

Three default databases of fragment-like molecules are provided with your SilcsBio software. You may choose to use one of these for your SILCS-Hotspots run, or use your own custom molecule or database of molecules. The databases are provided both as directories containing one molecule per Mol2 file and as a single SDF file containing all molecules in the database.

- `$SILCSBIODIR/data/databases/{ring_system,ring_system.sdf}`:
  Mono- and bicyclic rings commonly appearing in drug-like compounds [12].



- `$SILCSBIODIR/data/databases/{ring_subset,ring_subset.sdf}`: A small subset of the `ring_system` database.



- `$SILCSBIODIR/data/databases/{astex_mini_frag,astex_mini_frag.sdf}`: Polar fragments curated from Astex [13].

6.4 SILCS-Pharm

6.4.1 Background

Three-dimensional (3D) pharmacophore models (also called "hypotheses") offer an intuitive and powerful approach for virtual screening (VS). 3D pharmacophore VS works by searching for matches between the 3D pattern of functional groups constituting a pharmacophore model and the 3D arrangement of functional groups in ligand conformers in a virtual database. Ligands having conformers that closely match the pharmacophore model are considered hits. Traditionally, 3D pharmacophore models were developed using experimental knowledge of the binding poses for ligands to a receptor. This is in contrast to energy function-based VS ("docking"). An advantage of docking approaches was that they only require experimental knowledge of the receptor structure, and not of bound ligands. However, it is possible to develop pharmacophore models using only the

receptor structure. One means to generate such receptor-based pharmacophore models is with data from SILCS simulations.

SILCS-Pharm converts Grid Free Energy (GFE) FragMaps into 3D pharmacorphore models. GFE FragMaps from SILCS simulations are used as inputs for the four-step SILCS-Pharm process:

1. voxel selection;
2. voxel clustering and FragMap feature generation;
3. FragMap feature to pharmacophore feature conversion;
4. generation of a pharmacophore model for virtual screening (VS).

Here, "feature" refers to the identity and location of a chemical functional group and "pharmacophore model" is a collection of pharmacophore features.

FragMap generation from SILCS MD entails partitioning 3D space into uniform cubic voxels and enumerating fragment binding probabilities for each voxel. The voxels are retained during Boltzmann-inversion of probabilities to create GFE FragMaps. As the GFE voxel value represents the binding strength of a functional group at that specific location on the protein surface, the first step identifies the most favorable interactions based on a particular GFE cutoff value. In the second step, clustering is performed to group adjacent voxels, with each unique cluster becoming a FragMap feature. In the third step, the FragMap features are classified and converted into pharmacophore features. Finally, the pharmacophore features are prioritized using Feature GFE (FGFE) scores to create a SILCS 3D pharmacophore model [14]. In the current scheme, the program searches through five FragMap types: Generic Apolar, Generic Donor, Generic Acceptor, Positively Charged, and Negatively Charged. Additionally, instead of using a rigidly-defined protein surface to determine regions that ligands cannot sample because of protein volume occlusion, the SILCS Exclusion Map is used. The Exclusion Map has been validated to be a better alternative to more traditional representations of the occluded volume of the protein, and it explicitly accounts for protein flexibility.

6.4.2 Running SILCS-Pharm

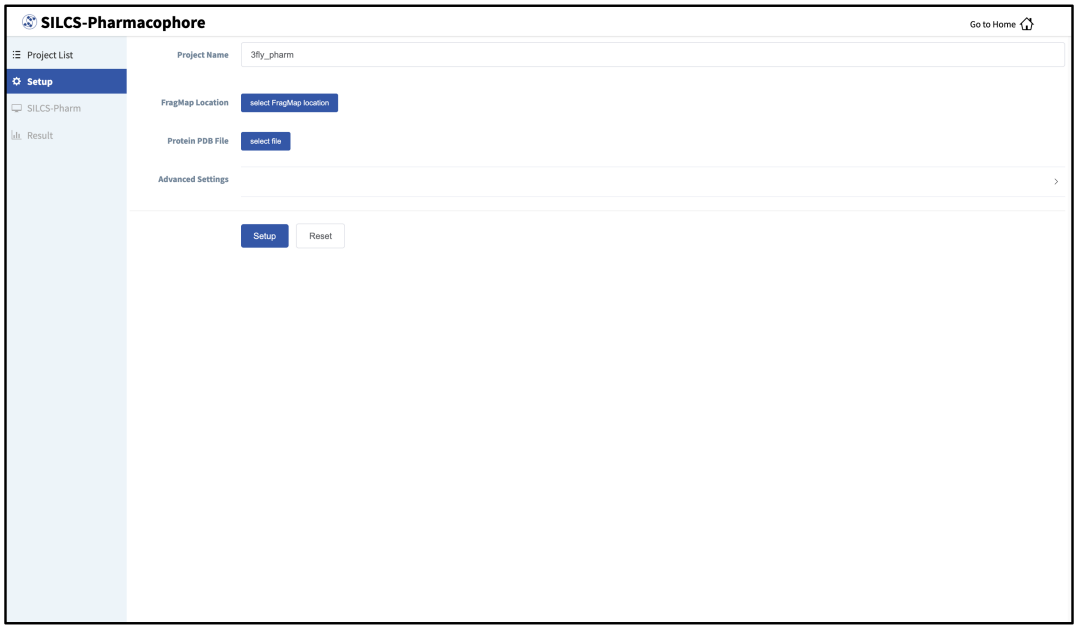
SILCS-Pharm Using the SilcsBio GUI

1. **Begin a new SILCS-Pharm project:**

   Select *New SILCS-Pharm project* from the Home page.

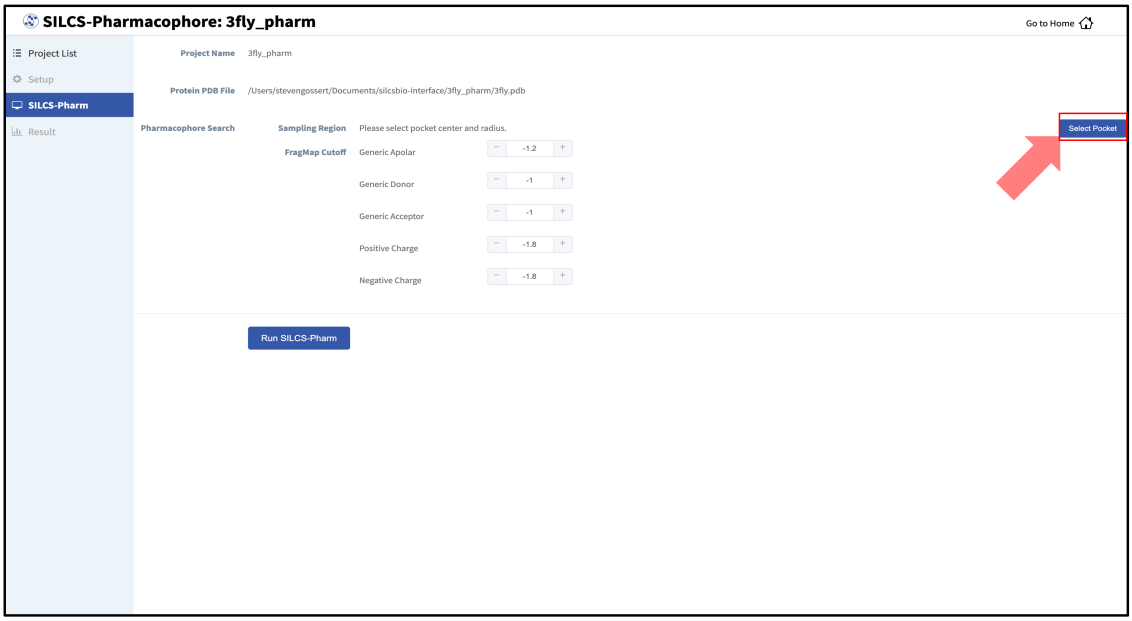
2. **Enter a project name and input files:**

   Enter a project name. Then, provide FragMap and protein input files. You may choose these files from the computer where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection*. Once all information is entered correctly, press the "Setup" button at the bottom of the page.



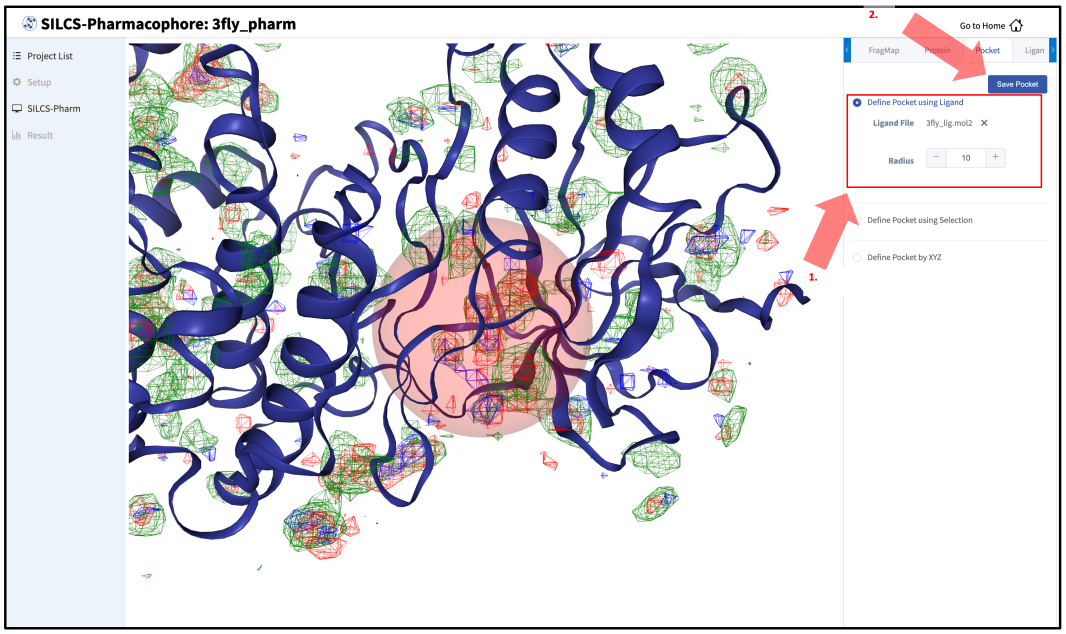
3. **Define the ligand binding pocket:**

   After pressing the "Setup" button at the bottom of the page, the screen will update to add "Pharmacophore Search" to the list of information. Click the "Select Pocket" button on the right-hand side of the window.



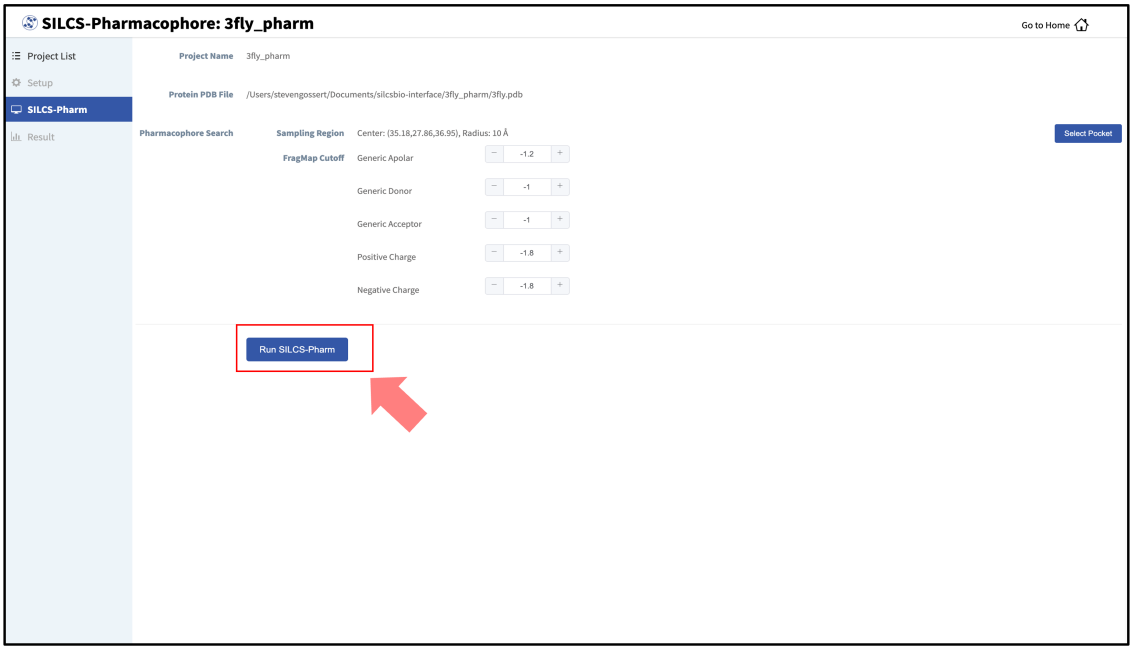
The GUI will now be showing the protein molecular graphic in the center pane. On the right-hand side, in the "Pocket" tab, you will need to define the pocket center based on the center-of-geometry of a ligand pose ("Define Pocket using Ligand"), or a target residue selection ("Define Pocket using Selection"), or by directly entering an x, y, z coordinate ("Define

Pocket by XYZ"). You will also need to choose a radius (default value "10") to complete the definition of the spherical pocket. If it is difficult to see the spherical pocket definition in the center pane, hide the protein surface representation. Click on the "Save Pocket" button and the "OK" acknowledgement to continue.

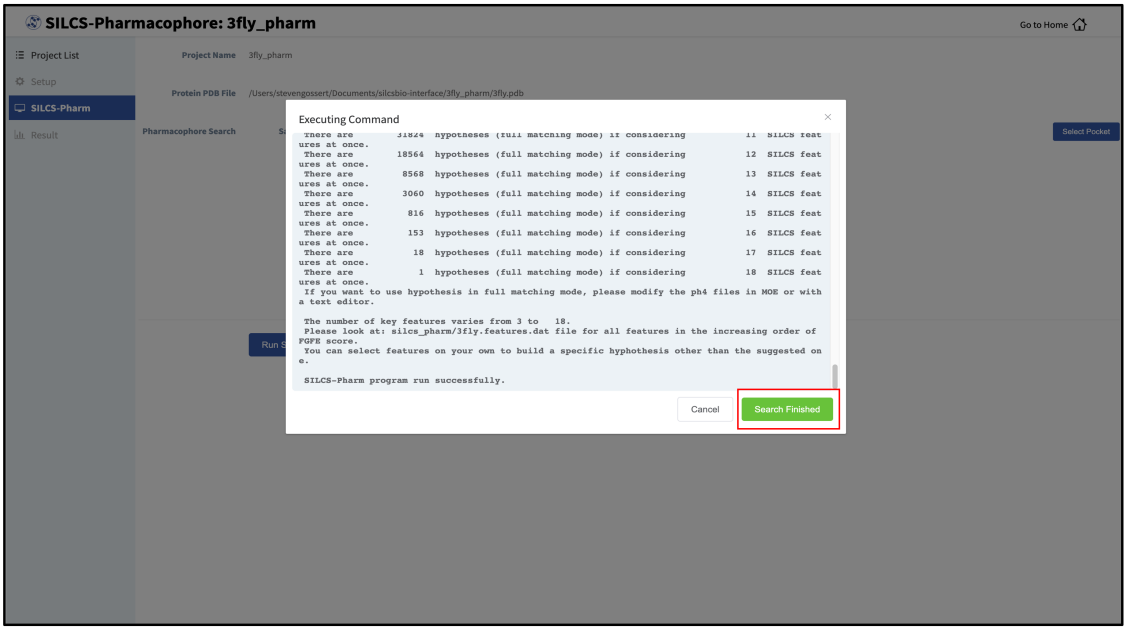


4. **Run SILCS-Pharm:**

   You will be returned to the previous screen, which now includes "Sampling Region" information consisting of the spherical pocket center and radius. You will also see default FragMap cutoff values listed for selection of FragMap densities for creation of pharmacophore features. You may adjust those values if you desire. Click on the "Run SILCS-Pharm" button to run the four-step SILCS-Pharm process.

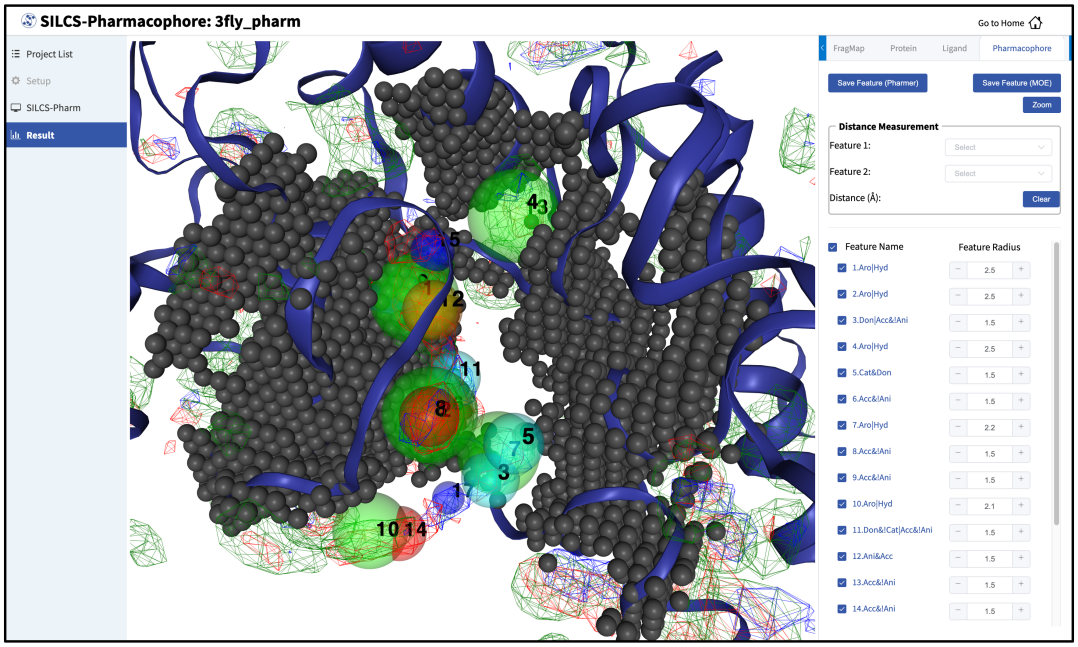


A pop-over window will appear and show job output, and the job will run to completion in a matter of seconds. Click the green "Search Finished" button.

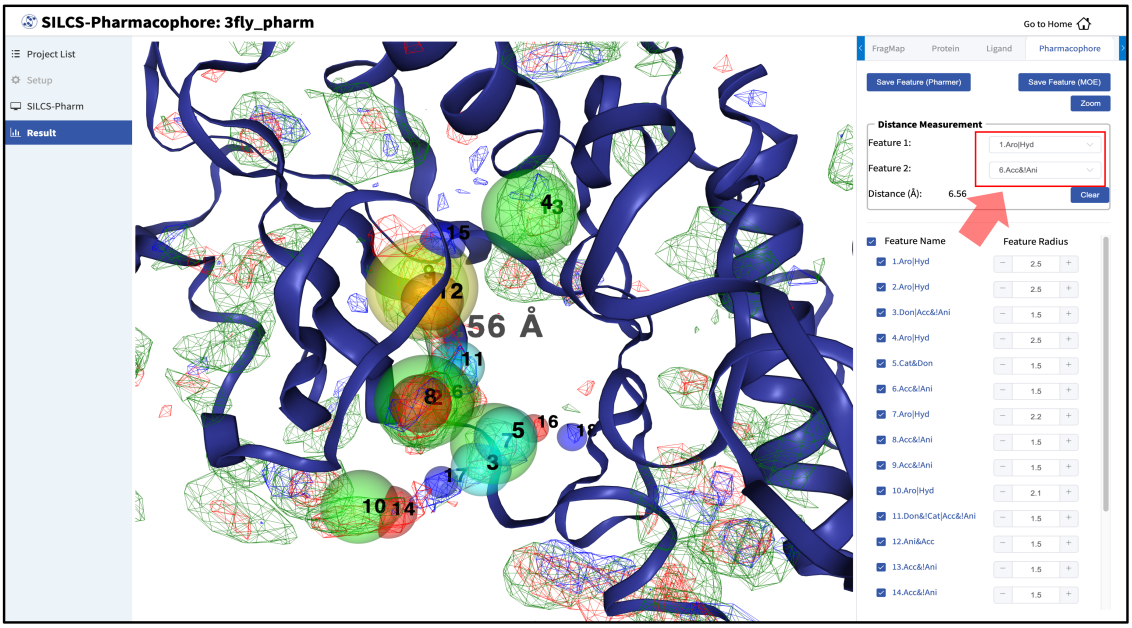


5. **Visualize and save pharmacophore models:**

   After clicking the green "Search Finished" button, the GUI will allow you to visualize the pharmacophore features on the protein. You may wish to click on the "Protein" tab in the right-hand panel and deselect the protein for a cleaner view of the pharmacophore features in the field of fragmaps.



The "Pharmacophore" tab will list all the pharmacophore features automatically generated from the FragMaps. You may adjust the radii of the individual features or deselect them entirely. You can also determine the distance between any two features to assist your analysis and choices.



Once finished with pharmacophore feature selection and adjustment, click the "Save Feature" button. You have the option to save the features in either Pharmer or MOE formats. This will allow you to save your pharmacophore model containing your selected/adjusted features in `.ph4` format on your local computer. By selecting/adjusting different subsets of the pharmacophore features, including the Exclusion Map, you may create multiple different pharma-

cophore models and save each one as a separate `.ph4` file. Note to run the pharmacophore screening you must select a minimum of three features and that exclusion pharmacophore features are NOT supported by Pharmer. This capability allows for different pharmacophore model screens of a binding site. The resulting diverse hits from those screens can be combined and, for example, subjected to SILCS-MC Pose Refinement for rescoring. Pre-existing pharmacophore models saved in `.ph4` format can also be viewed and edited through *View/Edit Pharmacophore File* on the "Home" page (see *View and Edit Pharmacophore Files* for more details).

SILCS-Pharm Using the CLI

A single command line interface command performs the four-step SILCS-Pharm process:

```
${SILCSBIODIR}/silcs-pharm/1_calc_silcs_pharm prot=<prot pdb> center="x,y,
↪z"
```

The input arguments are the PDB file used for the SILCS run and the absolute position of the center of a 10 Å sphere to be used to define the boundaries of the pharmacophore model. Two output files result from this command. `<prot>.keyf_<#features>.ph4` can be directly used for 3D pharmacophore VS by compatible programs (see below for generating Pharmer-compatible ph4 files). `<prot>_silcspharm_features.pdb` provides output in PDB format for easy visualization using standard molecular graphics packages. Running the command with no arguments

```
${SILCSBIODIR}/silcs-pharm/1_calc_silcs_pharm
```

will list additional options. Along with the required arguments of `prot=<prot pdb>` and `center="x,y,z"`, the following additional options can be set.

Optional parameters:

- FragMap directory path:

  ```
  mapsdir=<location and name of directory containing FragMaps;␣
  ↪default=maps>
  ```

  By default, the program will search for FragMaps in the `maps` directory.

- Output directory path:

  ```
  outputdir=<location and name of output directory; default=5_
  ↪pharm>
  ```

  By default, the program creates the directory `5_pharm` and places all output files there.

- Radius:

```
radius=<default: 10>
```

By default, a radius of 10 Å centered at `center="x,y,z"` is searched to generate pharmacophore features from the input FragMaps.

- Generic Apolar FragMap cutoff:

```
apolar_cutoff=<default: -1.2>
```

By default, Generic Apolar FragMap voxels having a GFE value <= -1.2 kcal/mol are selected.

- Generic Donor FragMap cutoff:

```
hbdon_cutoff=<default: -1.0>
```

By default, Generic Donor FragMap voxels having a GFE value <= -1.0 kcal/mol are selected.

- Generic Acceptor FragMap cutoff:

```
hbacc_cutoff=<default: -1.0>
```

By default, Generic Acceptor FragMap voxels having a GFE value <= -1.0 kcal/mol are selected.

- Methylammonium N FragMap cutoff:

```
mamn_cutoff=<default: -1.8>
```

By default, Methylammonium N FragMap FragMap voxels having a GFE value <= -1.8 kcal/mol are selected.

- Acetate O FragMap cutoff :

```
aceo_cutoff=<default: -1.8>
```

By default, Acetate O FragMap FragMap voxels having a GFE value <= -1.8 kcal/mol are selected.

In addition to visualization, the output PDB file `<prot>_silcspharm_features.pdb` can be used for easy editing of the pharmacophore model. Using a text editor, modify/reduce the features in this file as desired and save the revised file as `<prot>_silcspharm_features_revised.pdb`. Using this revised PDB file and the original ph4 file `<prot>.keyf_<#features>.ph4` as input, create a new ph4 file with:

```
${SILCSBIODIR}/programs/revise_ph4 <prot>.keyf_<#features>.ph4 <prot>_
↪silcspharm_features_revised.pdb
```

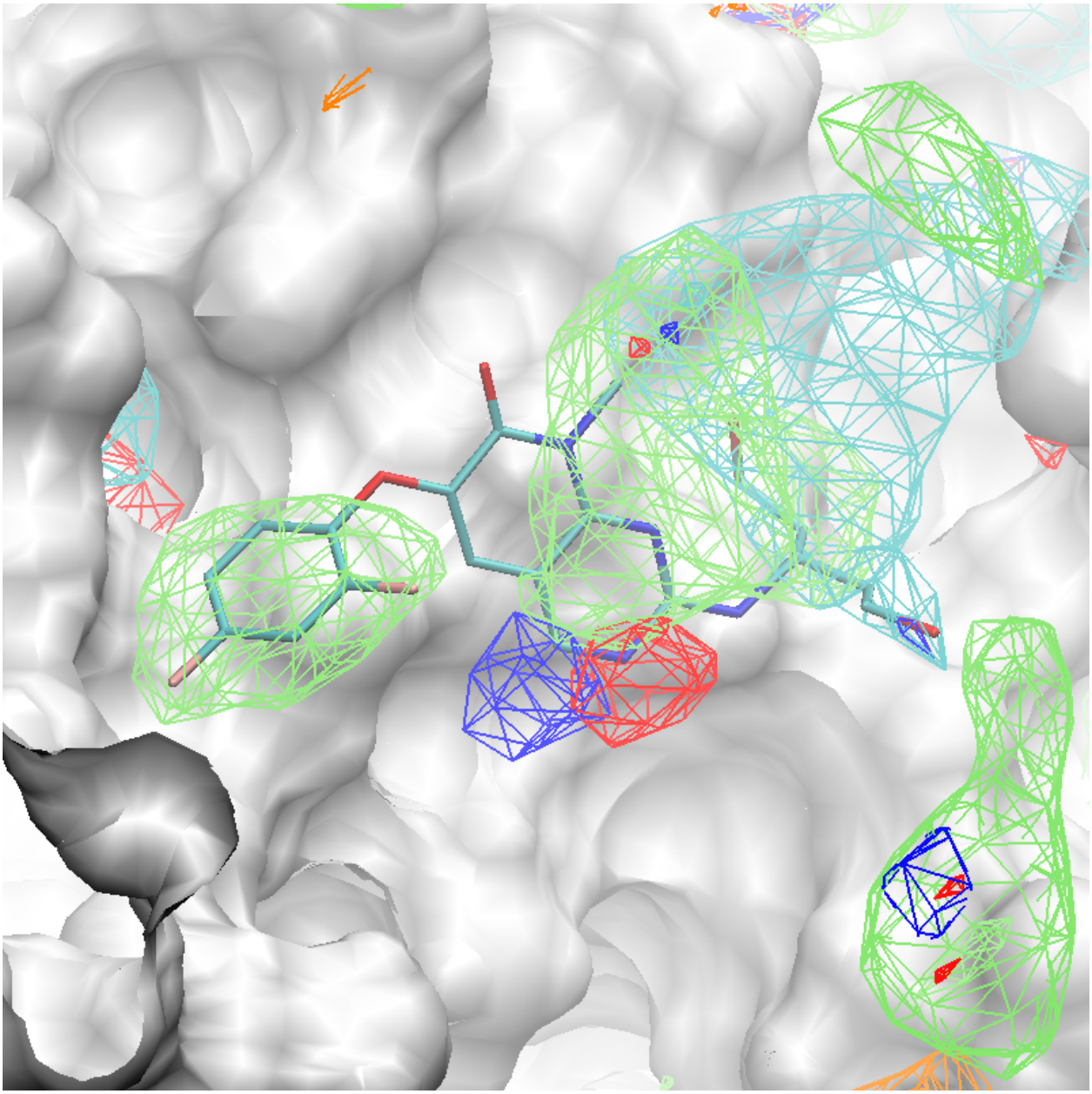
The output of this command will be the revised ph4 file <prot>.keyf_<#features>_revised.ph4. To create Pharmer-compatible ph4 output, add the pharmer option:

```
${SILCSBIODIR}/programs/revise_ph4 <prot>.keyf_<#features>.ph4 <prot>_
↪silcspharm_features_revised.pdb pharmer
```

Example Case Using the CLI

The following example demonstrates use of SILCS-Pharm from the command line interface to generate a pharmacophore model for p38 MAP kinase. Input files, including FragMaps, are provided in ${SILCSBIODIR}/examples/silcs/ (these same input files may also be used for a trial run of SILCS-Pharm with the SilcsBio GUI).

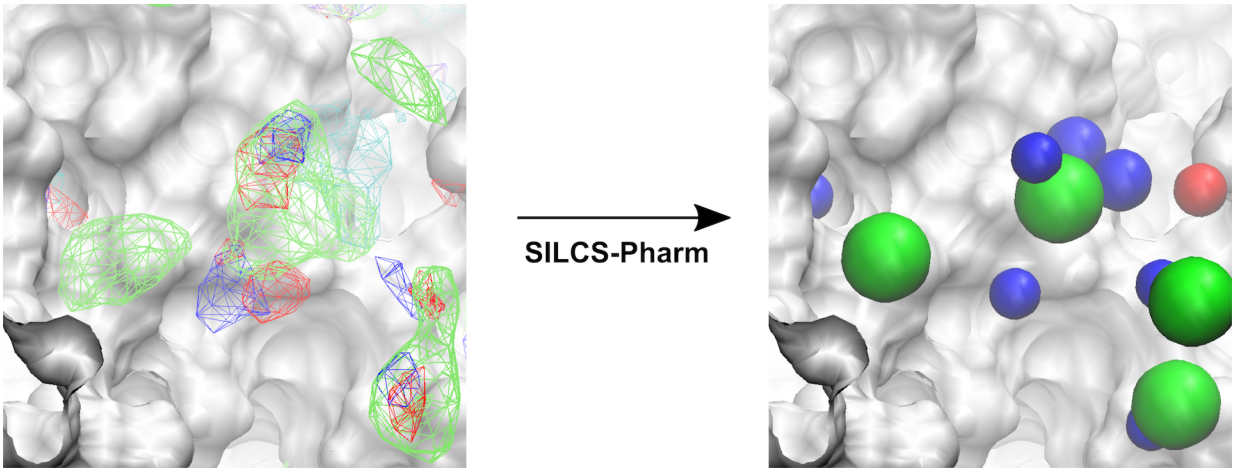
The x,y,z coordinates of the center of the complexed ligand are used to create a SILCS-Pharm 3D pharmacophore model encompassing the ligand binding site (shown below). These coordinates define the center of the sphere within which FragMap voxels are searched and clustered to generate features.



Using the ligand center coordinates `center="35.24, 27.48, 37.73"`, generate the 3D pharmacophore model with:

```
${SILCSBIODIR}/silcs-pharm/1_calc_silcs_pharm prot=3fly.pdb center="35.24,
↪27.48,37.73" mapsdir=${SILCSBIODIR}/examples/silcs/silcs_fragmaps_3fly/
↪maps
```

The command will complete after several seconds, and the output will note, "A total of 13 features have been detected." All output files will be in a new subdirectory `5_pharm`. Standard molecular graphics software can be used to visualize the output file `3fly_silcspharm_features.pdb`, which has one `ATOM` entry for each of the 13 pharmacophore features:

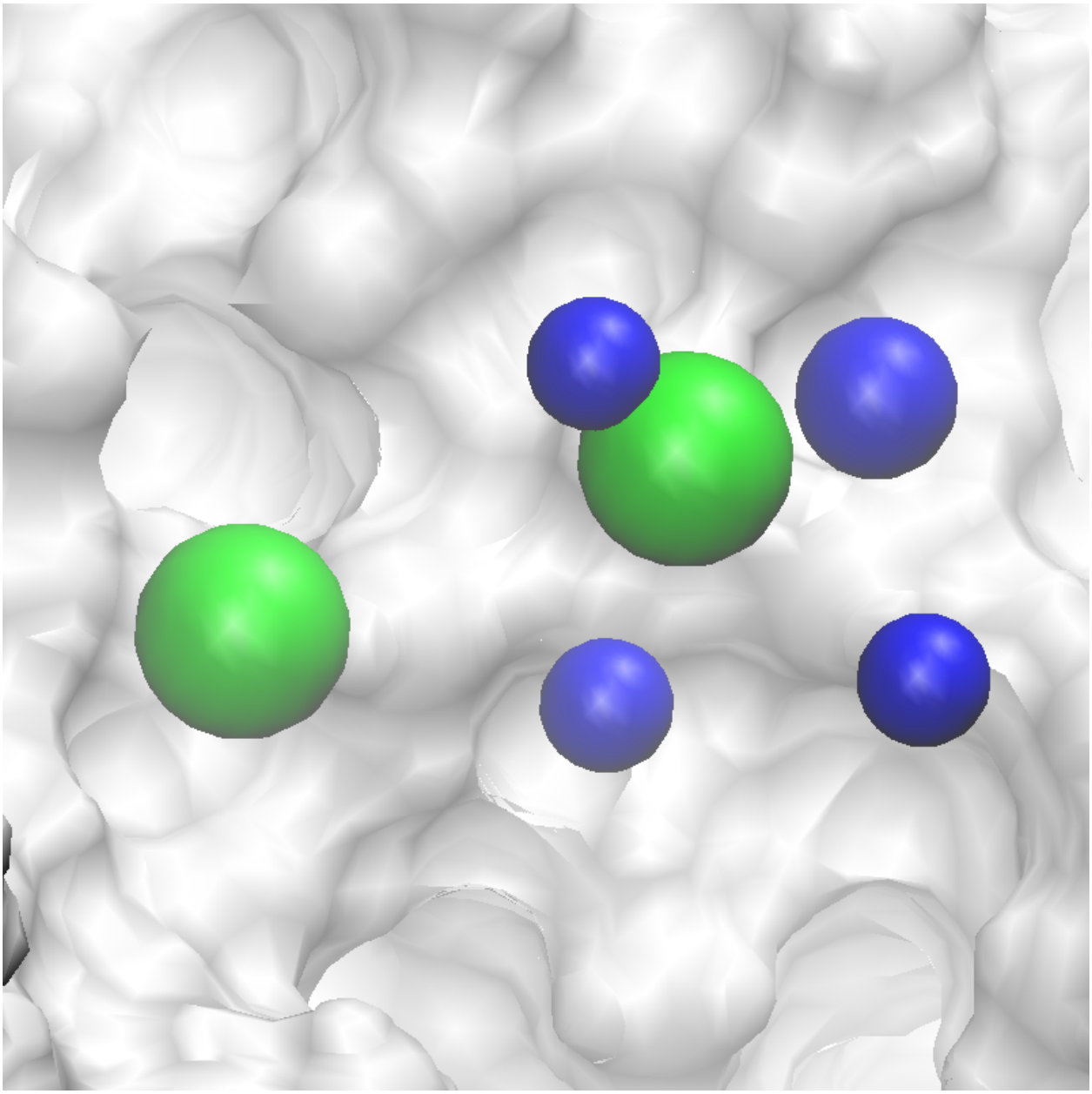


The conversion of FragMaps (left) to SILCS pharmacophore features (right) is shown above.

To modify/reduce the features, edit the output file `3fly_silcspharm_features.pdb` and save it as `3fly_silcspharm_features_revised.pdb`, then run `${SILCSBIODIR}/programs/revise_ph4`:

```
${SILCSBIODIR}/programs/revise_ph4 3fly.keyf_13.ph4 3fly_silcspharm_
↪features_revised.pdb
```

The output `3fly.keyf_13_revised.ph4` will reflect your revisions in `3fly_silcspharm_features_revised.pdb`. Below are the revised SILCS pharmacophore features.



6.4.3 Pharmacophore Screening with Pharmer

Through the integration of Pharmer, an open-source pharmacophore search program, into the SilcsBio software, users are able to perform high-throughput virtual screening (HTVS) using pharmacophore features generated by SILCS-Pharm. To perform HTVS with Pharmer, please follow the steps detailed in this section.

1. **Check Pharmer installation:**

Make sure that your Pharmer program is installed correctly using the following command:

```
$SILCSBIODIR/programs/pharmer --help
```

If the above command produces an error, please see *PHARMER Installation*.

2. **Check database imports:**

High-throughput virtual screening (HTVS) requires a prepared database of screening compounds. SilcsBio users can get *Ready-To-Screen* Pharmer compatible databases that include multiple protomers and conformers of each screening compound upon request. Please contact support@silcsbio.com to request a prepared database. Alternatively, users may also choose to build their databases locally. Instructions on how to build a database are provided in *Building a Pharmer-Compatible Database*.

Once the database has been prepared, ensure that the database is linked to `SILCSBIODIR` using the following command:

```
ls $SILCSBIODIR/data/databases/pharmer_databases
```

The above command will output a list of available databases.

If the above command results in an empty output or the desired database is not listed, then please follow the steps below to link a database (e.g., `molport`) to `SILCSBIODIR`.

```
# create the directory for your database
mkdir -p $SILCSBIODIR/data/databases/pharmer_databases/molport

# create symbolic links for parts of your prepared database
ln -s Part_* $SILCSBIODIR/data/databases/pharmer_databases/molport/
```

Note that the directory being used for the database should contain the individual parts of the database (e.g., `Part_1`, `Part_2`, etc.) and not the SDF file used to build the database or a sub-directory.

3. **Run Pharmer to perform virtual screening:**

With both Pharmer installed and the appropriate database(s) linked, users can perform HTVS on a pharmacophore model using the command below:

```
$SILCSBIODIR/silcs-pharm/2_run_pharmer ph4=<PH4 file> database=<database␣
↪to screen>
```

Required parameters:

- Path and name of the input pharmacophore model in `.ph4` format:

  ```
  ph4=<PH4 file>
  ```

  The `.ph4` file containing your pharmacophore model must be in Pharmer format. If you do not have a `.ph4` file, please refer to *Running SILCS-Pharm* to create one.

- Name of the database to screen:

```
database=<database to screen>
```

  You may include multiple databases in your virtual screening project by providing space-separated names (e.g., `database="molport enamine"`).

Optional parameters:

- Name of the directory containing the output:

```
outputdir=<location and name of output directory; default=5_
↪pharm>
```

- Path to the Pharmer program executable:

```
pharmer=<location and name of Pharmer executable; default=
↪$SILCSBIODIR/programs/pharmer>
```

- Number of processors available for running the Pharmer program:

```
nproc=<# of cores available to Pharmer; default nproc=8>
```

  The default number of processors used is determined when the SilcsBio Software is set up on the server on which the calculations are performed.

```
batch=<determines if the Pharmer jobs are automatically submitted; true/
↪false; default=false(submits automatically)>
```

Setting `batch=true` will create job scripts for each part of the database to be screened, but will not automatically submit those jobs. This allows users to review the job scripts before submission. If you set `batch=true`, you will need to submit the jobs manually using the generated job scripts in the output directory.

Warning: The Pharmer program does not support pharmacophore features representing the Exclusion Map.

Note: It is imperative that the selected database(s) is a conformational database, which includes a representative set of conformers for each molecule. Pharmer screening entails only rigid translations and rotations of the entire molecule; **NO** internal rotations of torsion angles are performed. For assistance in creating a custom conformer database, please contact us at support@silcsbio.com.

6.4.4 Building a Pharmer-Compatible Database

Pharmer requires a database of molecules to screen. The database should be in Pharmer format, which includes multiple protomers and conformers of each screening compound. The database should also be in a directory structure that is accessible to the computer on which `2_run_pharmer` command (step 3 under *Pharmacophore Screening with Pharmer*) will be run.

To build a Pharmer-compatible database, follow the steps below:

1. **Collect structures of all molecules of interest in one SDF file:** Prepare an SDF file containing all of the molecules you want to screen. Make sure to enumerate protomers for each molecule. Alternately, those with access to CCG's MOE will have the option to use MOE (`sdwash`) to wash and enumerate protomers.

Tip: Note that the SDF file may contain a property with unique identifier for each molecule. The property name can be used to rename the molecules in the database.

---

2. **Build the Pharmer-compatible database:**

After preparing the SDF file containing all molecules of interest, run the `build_pharmer_database` command:

```
$SILCSBIODIR/silcs-pharm/build_pharmer_database dbname=<database name>␣
↪sdfile=<location and name of SDF file of the compounds>
```

Required parameters:

- Name of the output database (e.g. `molport_2024_01`):

  ```
  dbname=<database name>
  ```

  Please use alphanumeric characters, dashes, and underscores only. The database name will be used to create the directory structure for the database.

- Path and name of SDF file of the compounds:

  ```
  sdfile=<location and name of SDF file of the compounds>
  ```

  This is the SDF file containing the molecules to be screened. The SDF file can have 2D or 3D coordinates. No hydrogen atoms are required in the SDF file as they will be added by RDKit.

Optional parameters:

- Number of conformers to generate:

  ```
  nconformers=<# of conformers to generate; default=10>
  ```

The default number of conformers to generate is `10`. An RMSD threshold of 0.35 Å is used to filter out similar conformers.

- Number of molecules per part:

```
molsperpart=<# of molecules per part; default=50000>
```

The output database will be divided into parts, with each part containing a specified number of molecules. The default number of molecules per part is `50,000`. The number of parts will be determined by the number of molecules in the database. The parts will be named `Part_1`, `Part_2`, etc.

- The number of the first part:

```
firstpart=<first part number; default=1>
```

The default number of the first part is `1`. This option is used to specify the number of the first part in the database. This may be useful if you are appending new parts to an existing database. This may also be useful if you are building a large database requiring the use of different storage locations for additional parts.

- Unique identifier property in SDF file:

```
molid=<unique identifier property in SDF file to set␣
↪molecule name>
```

The value of this property will be used to rename the molecules in the database during the washing stage. Combined with the `prefix` parameter, users can use this option to rename the molecules in the database as `<prefix>_<molid>`. For example, `molid="PUBCHEM_EXT_DATASOURCE_REGID"` is a unique identifier property for the SDF files of the MolPort database.

- Prefix for the molecule names:

```
prefix=<a prefix for the molecule name>
```

The specified prefix will be added to the molecule name in the database. If you used the `molid` parameter, the molecule name will be the value of the property in the SDF file with the prefix added.

- Skip the washing and enumeration of protomers:

```
skipwash=<skip washing and protomer generation; true/false;␣
↪default=false>
```

If you have already washed and enumerated protomers for the molecules in the SDF file, you can skip this step by setting `skipwash=true`. This option can NOT be used with the `moewash` option. Also, this option should NOT be used if you want to use `molid` and `prefix` options.

- Number of cores available to Pharmer:

```
nproc=<# of cores available to Pharmer; default=8>
```

- Path to the Python executable for RDKit:

```
python=<path to python executable for rdkit; default=$(which␣
↪python)>
```

- Add newly created database to default path:

```
addtopath=<move the database to pharmer_databases directory;␣
↪true/false; default=false>
```

  Use this option if you want to make the database available in the default `dbpath` for pharmacophore screening with the `2_run_pharmer` script. Setting this option to `true` will copy the database to the `$SILCSBIODIR/data/databases/pharmer_databases` directory. Alternative options are:

  1. Create a symbolic link of the database directory in the default `dbpath` (i.e., `$SILCSBIODIR/data/databases/pharmer_databases`) directory. Note that the original database should be accessible from the compute node where the screening job scripts (`job_pharmer*cmd`) get submitted. Individual parts of the database could also be linked separately if certain parts are stored in different locations. It is okay if the symbolic links are inactive on the login node.
  2. Use the `dbpath` option in the `2_run_pharmer` script to specify the location of the database. Again, the database should be accessible from the compute node where the screening job scripts (`job_pharmer*cmd`) get submitted.

Additional parameters:

- Use MOE by CCG to wash and enumerate protomers:

```
moewash=<use MOE to wash and enumerate protomers; true/false;
↪ default=false>
```

- Path to the MOE (sdwash) executable for washing:

```
sdwash=<path to MOE(sdwash) executable for washing; default=>
```

- Number of protomers to generate:

```
nprotomers=<# of protomers to generate; default=3>
```

- Number of MOE tokens available to use:

```
moetokens=<number of MOE tokens available; default=3>
```

6.5 SILCS-MC: Docking and Pose Refinement

6.5.1 SILCS-MC Background

The power of SILCS lies in the ability to use FragMaps to rapidly evaluate binding of diverse ligands to a target. SILCS-MC is Monte-Carlo (MC) sampling of ligands in translational, rotational, and torsional space in the field of the FragMaps and Exclusion Map. MC sampling uses CGenFF force field intramolecular energies, the Ligand Grid Free Energy (LGFE), which is the sum of atomic Grid Free Energies (GFEs), and the Exclusion Map. The Exclusion Map prevents ligand sampling where no probe or water molecules visited during SILCS simulations. SILCS-MC allows for rapid conformational sampling of the ligand while accounting for protein flexibility in a mean-field-like fashion since ligand affinity and volume exclusion are embedded in the combination of FragMaps and the Exclusion Map. Final ligand scoring is based on the LGFE score [15] [16]. In addition, the atomic GFE scores that comprise the LGFE score represent the free energy contribution of individual atoms to binding. This information can greatly facilitate the interpretation of experimental data and ligand design.

SILCS-MC can be used to generate and score binding poses for a ligand to a target using SILCS FragMaps that have been previously computed for that target (see *SILCS Simulations*). This can readily be done for a single ligand or a database of ligands. Two default conformational sampling protocols are available, "docking" and "pose refinement," as described in detail below. The docking protocol is useful when no information is available about the binding pose, as it entails extensive translational, rotational, and intramolecular conformational sampling. The pose refinement protocol is useful when a reasonable starting pose for the ligand is available, for example to re-score poses output by another high-throughput in silico screening tool. Instructions for developing custom SILCS-MC ligand conformational sampling protocols are provided at the end of this chapter.

6.5.2 SILCS-MC Docking

Docking is exhaustive sampling of a ligand's conformation in a given pocket to determine its most favorable orientation and internal geometry as defined by LGFE scoring. The pocket for the purposes of ligand sampling is defined as a sphere with a given radius and center; the same pocket is used for all ligands in a given SILCS-MC run. This protocol entails one MC run with the ligand for GPU jobs and five independent MC runs with the ligand for CPU jobs.

This protocol is recommended for ligands with diverse chemotypes and unknown binding poses. When the pose of a parent ligand is known and SILCS-MC evaluations are to be performed over a congeneric series, the pose refinement protocol is recommended instead (see below).

SILCS-MC Docking Using the SilcsBio GUI

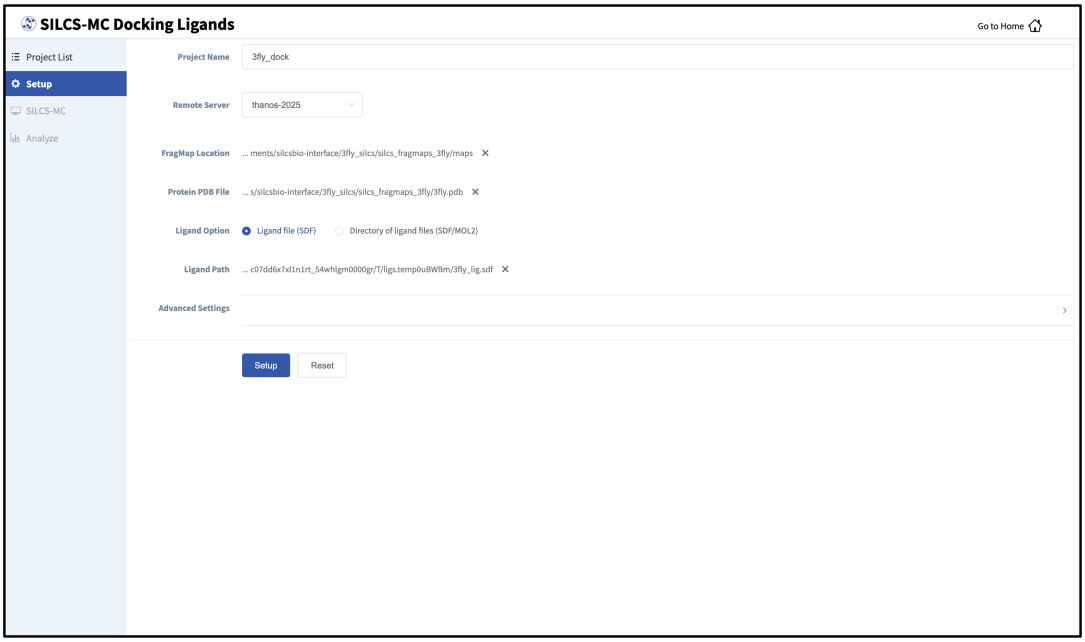
1. **Begin a new SILCS-MC Docking project:**

   Select *New SILCS-MC Docking project* from the Home page.

2. **Enter a project name, select the remote server, and input files:**

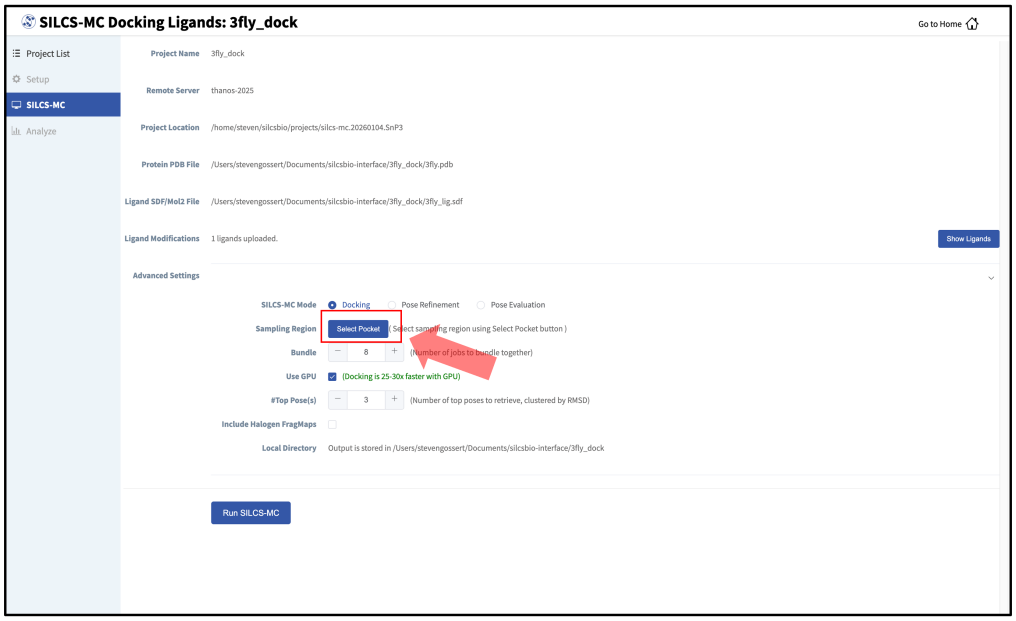
   Enter a project name and select the remote server where the SILCS-MC docking jobs will run. Also, provide FragMap and protein input files. You may choose these files from the computer where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection*. You will additionally need to provide a "Ligand file" or "Directory of ligand files" that contains the database of ligands to be docked. Note that users can modify ligands using the "Modify Ligand" utility available on the "Home Page" (see *Modify Ligand for SILCS-MC*).

   Warning: Ligands in the "Ligand file" or "Directory of ligand files" must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.



3. **Select the SILCS-MC sampling protocol:**

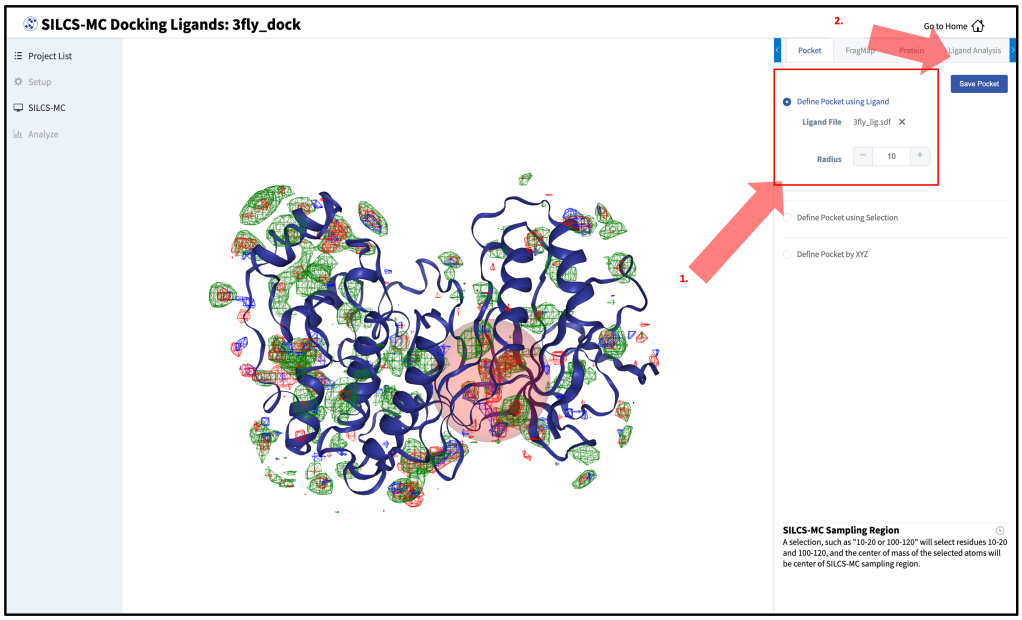
   Once all information is entered correctly, press the "Setup" button at the bottom of the page. The page will update to list the number of ligands and show options for the sampling protocol ("Docking", "Pose Refinement", or "Pose Evaluation") and the sampling region. Select the "Docking" option and then press the "Select Pocket" button.



4. **Define a SILCS-MC sampling region:**

   The GUI will now be showing the protein molecular graphic in the center pane. On the right-hand side in the "Pocket" tab, you can define the pocket center based on the center-of-geometry of a ligand pose ("Define Pocket using Ligand"), or a target residue selection ("Define Pocket using Selection"), or by directly entering an x, y, z coordinate ("Define Pocket by XYZ"). You may also adjust the sampling region's radius (default value "10" Å) by using the "+/–" buttons. If it is difficult to see the spherical pocket definition in the center panel, hide the protein surface representation. Click on the "Save Pocket" button and the "OK" acknowledgement to continue.

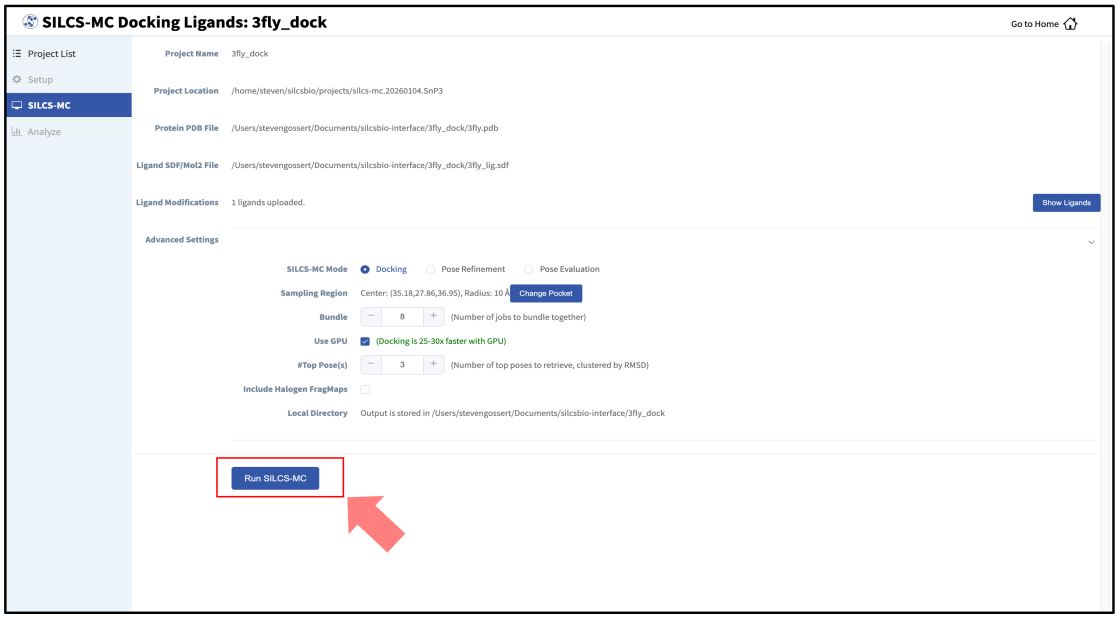
   Warning: Adjusting the radius of the spherical pocket may negatively impact the SILCS-MC sampling. Using a radius smaller than the default 10 Å reduces the size of the sampling region, which may disallow the generation of sufficient number of conformations leading to the optimal binding pose being missed. Using a radius larger than the default 10 Å increases the size of the sampling region, thereby increasing the computational cost and time of the SILCS-MC docking runs and increasing the number of irrelevant binding poses.



5. **Launch SILCS-MC jobs:**

   You will be returned to the previous screen, which now includes "Sampling Region" information consisting of the spherical pocket center and radius.

Tip: If you ran Halogen SILCS simulations for your target, you can include the Halogen SILCS FragMaps in the SILCS-MC posing and scoring by checking the "Include Halogen FragMaps" box.

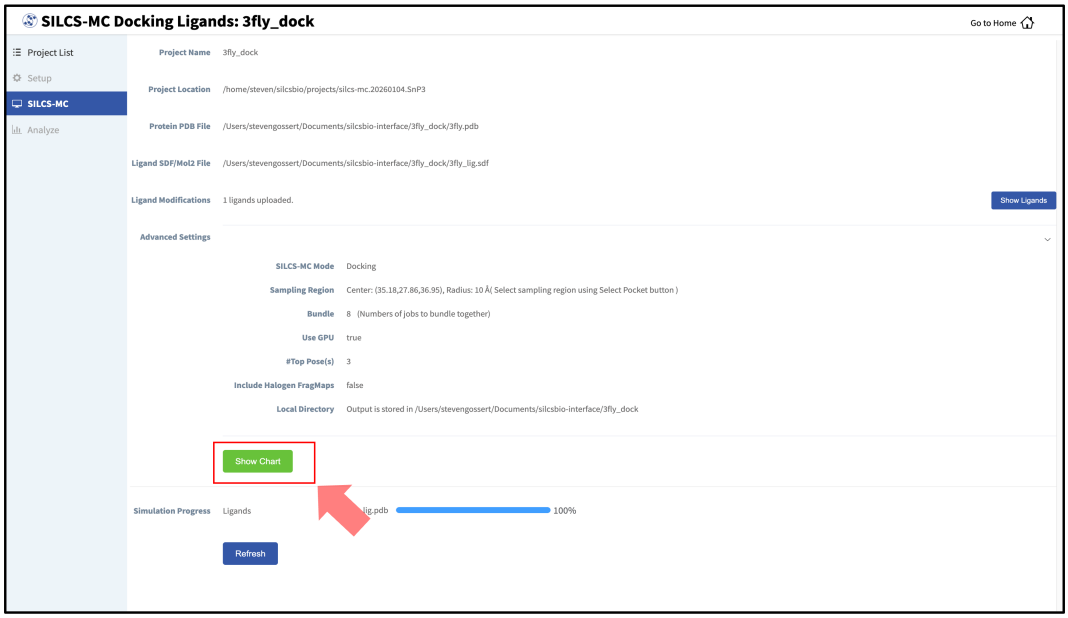


You may now click the "Run SILCS-MC" button to start the SILCS-MC docking. Doing so will submit jobs to the remote server job schedular and list them in a pop-over window.

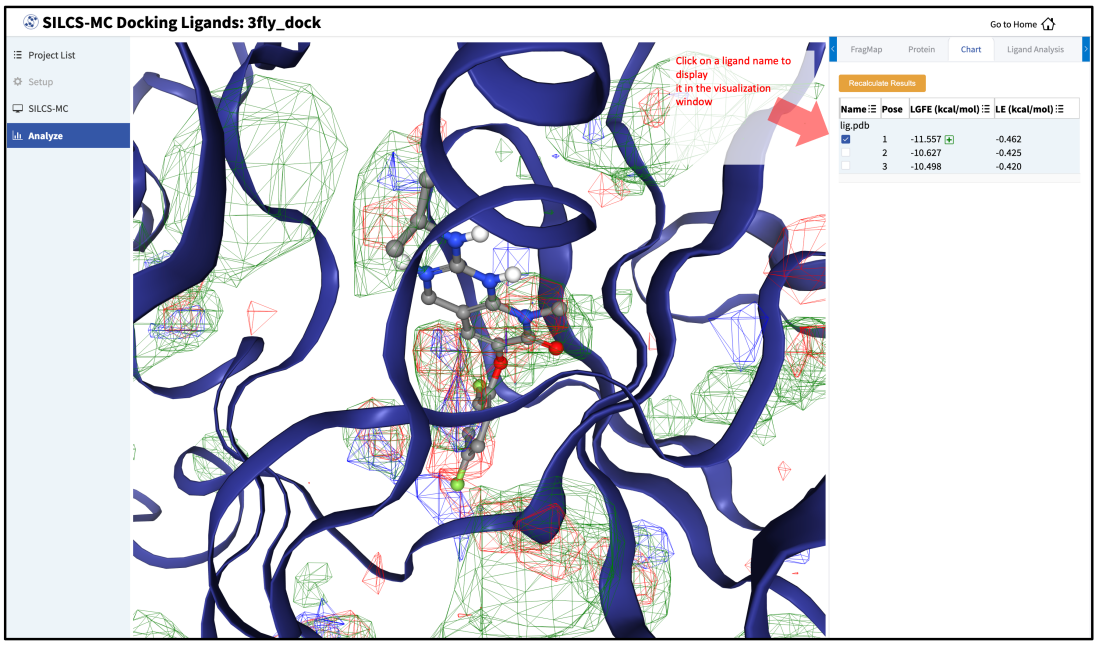
Once all jobs are submitted, you may click on the "Setup Successful" button to dismiss the pop-over window and return to the previous screen, which will now show a "Simulation Progress" section. You may update this section by scrolling to the bottom of the screen and clicking the "Refresh" button. This will update the progress bars for all of the ligands being docked.

6. **Visualize SILCS-MC docked poses:**

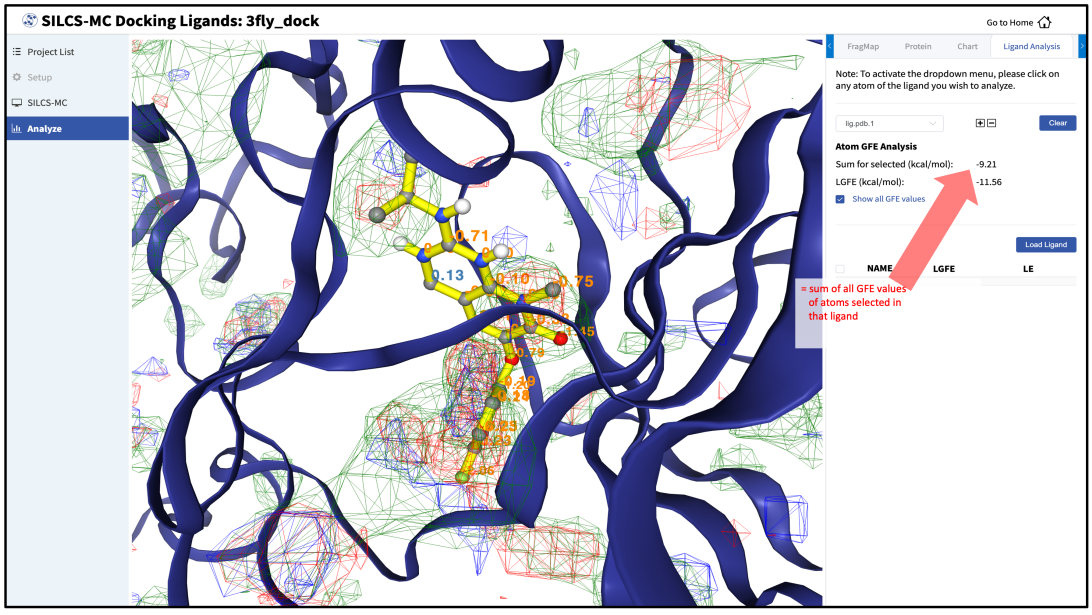
   Once progress bars for all ligands reach 100%, click on the "Show Chart" button above the "Simulation Progress" section to proceed.



Upon successful completion of this command in the pop-over window, you may click on the green "Data collection finished" button to return to the GUI. A new tab, labeled "Chart" will have been created in the right-hand panel. Under that tab, clicking on a ligand name will display that ligand.



Choosing the "GFE" tab in the right-hand panel will allow you to click on individual atoms within a ligand to see their atomic GFE values. The "GFE" tab will also display the sum of all GFE values of atoms selected in that ligand.



SILCS-MC Docking Using the CLI

Ligands can be docked using SILCS-MC docking through the following steps:

1. **Launch SILCS-MC docking runs:**

   To set up and run SILCS-MC docking from the command line interface, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate SDF or Mol2 file. Alternatively, all the ligands can be combined into a single SD file. With this information, enter the following command to set up and launch SILCS-MC docking runs:

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_exhaustive prot=<prot pdb>␣
↪ligdir=<ligand mol2/sdf directory>
```

   Required parameters:

   - Path and name of protein PDB file:

```
prot=<protein pdb file>
```

   - Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<location and name of directory containing ligand mol2/
↪sdf>
```

     If the ligdir parameter is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` parameter.

> **Warning:** Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

   Optional parameters:

   - Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

   - Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

   - Center of the spherical sampling region:

```
center=<"x,y,z"; e.g., center="20,30,-2"; default=ligand's␣
↪center of mass>
```

If the center is not specified by the user, the default center will be calculated from the input ligand's center of mass, and the initial conformation of the ligand in the SILCS-MC docking runs will correspond to the conformation of the ligand in the input Mol2 or SD file. If the center is specified, then the conformation and orientation of the ligand will be randomized prior to the SILCS-MC docking runs.

- Radius of the spherical sampling region:

```
radius=<SILCS-MC calculation radius from center in Å; default=10.
↪0>
```

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will overwrite␣
↪ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

- Alignment to reference molecule prior to SILCS-MC docking:

```
refsdf=<location and name of SD file of reference molecule to␣
↪align to; default=None>
```

If the `refsdf` option is used, the ligands in `sdfile` will be aligned to the reference molecule before the SILCS-MC docking runs are initiated. The alignment is performed using the Open3DAlign(O3A) algorithm [17] available in RDKit. Currently, this option is only available using the `sdfile` option, and is not available using the `ligdir` option.

- Inclusion of halogen FragMaps:

```
halogen=<use halogen maps; true/false; default=false>
```

If halogen FragMaps have been generated (see *Setup with halogen probes*), they can be included in the SILCS-MC calculation to improve the scoring of halogen-containing compounds. To do so, add the `halogen=true` option.

- Atom classification scheme (ACS):

```
class=<atom classification scheme (generic or specific);␣
↪default=generic>
```

Atomic GFEs are assigned based on the atom's classification and its overlap with the corresponding FragMap. The atom's classification is assigned based on the ACS used.

Users can choose between generic and specific ACS using the `class` option. The default ACS is the `generic` ACS in which the generic apolar, donor, and acceptor maps are used to score corresponding atoms. In the `specific` ACS, specific maps (e.g., formamide nitrogen FragMaps and imidizole donor nitrogen FragMaps are kept separate and are not grouped into generic donor FragMaps) to score corresponding atoms. The schemes are stored in `${SILCSBIODIR}/data/silcs/`. Further details regarding ACS and how they affect atomic GFEs and LGFEs are provided in ref. [18].

- Option to bundle jobs:

```
bundle=<bundle multiple (single) jobs into larger jobs; true/
↪false; default=false>
```

When `bundle=true` multiple (single) jobs will be bundled into a single larger job. The number of jobs to be bundled can be set with the `nproc` parameter.

- Number of jobs to bundle (when `bundle=true`):

```
nproc=<number of jobs to bundle; default=8>
```

By default number of jobs to bundle is determined by the number of CPU cores on the compute nodes available to the user. The user can override this using the `nproc` parameter.

- Path to python executable:

```
python=<path to python executable; default=$(which python)>
```

The default python path will be checked using the `which python` command.

- Use GPU version of SILCS-MC:

```
gpu=<use GPU; true/false; default=true>
```

By default, the GPU version of SILCS-MC is used. To use the CPU version, use `gpu=false`. The GPU version of SILCS-MC is only available on computers with NVIDIA GPUs and CUDA version 11 or later. The CPU version of SILCS-MC is available on all machines, however it is significantly slower than the GPU version. When using the CPU version of SILCS-MC, the SILCS-MC calculation will be split into five independent, single-core serial jobs per ligand. Also note that the `nproc` parameter must not be changed from its default value when using the `gpu=false bundle=true` options to bundle CPU jobs.

SILCS-MC Docking with multiple ligands:

Users may dock multiple ligands to a binding site simultaneously. To dock multiple ligands simultaneously, the user can either provide an additional directory containing all of the additional ligands to be docked as separate files in Mol2 or SD file format or an additional single SD file containing all of the ligands to be docked. These additional ligands, herein referred

to as secondary ligands, are docked in the same pocket as the primary ligand. To perform SILCS-MC docking with multiple ligands, the user can set the following parameters to dock the primary ligand with secondary ligands simultaneously:

Secondary Ligand Parameters:

- Perform SILCS-MC docking with secondary ligands:

```
seclig=<true/false; default=false>
```

  By default, the SILCS-MC docking will be performed with the primary ligand only. Users can perform SILCS-MC docking on multiple ligands simultaneously by setting `seclig=true`. When set to true, SILCS-MC will be performed on both the primary ligand (specified by `ligdir` or `sdfile`) and the secondary ligand. By default, 15 water molecules will be used as secondary ligands, however, the secondary ligand may also be specified with `secligdir` or `secsdfile` (described below).

- Path and name of directory containing secondary ligand SD or Mol2 files:

```
secligdir=<path to directory containing secondary ligand mol2/
↪sdf files>
```

  When `seclig=true`, by default the `secligdir` option is internally set to `${SILCSBIODIR}/templates/silcs-mc/secligdir`, which contains 15 water molecules. Users can provide a different set of secondary ligands by setting the `secligdir` option to the path of the directory containing the desired secondary ligand Mol2 or SD files.

Tip: To increase or decrease the number of water molecules simultaneously docked with the primary ligand, the user can copy the desired number of water molecules from the default `secligdir` directory (`${SILCSBIODIR}/templates/silcs-mc/secligdir`) to a new directory and set the `secligdir` option to the new directory. It does not matter if the coordinates of two water molecules are the same, as long as the file names are different.

- Path and name of secondary ligand SD file:

```
secsdfile=<path to sdf file of the secondary ligands; this␣
↪option will overwrite secligdir>
```

  If the `secsdfile` option is used, then the `secligdir` option is not needed and any input used for the `secligdir` option will be overwritten. The `secsdfile` option is recommended if the user has an SD file containing all secondary ligands to be used.

- Radius of the spherical sampling region for secondary ligands:

```
secradius=<MC-SILCS calculation radius from center for secondary␣
↪ligands; default=10.0>
```

When `seclig=true`, the default radius for the secondary ligands is set to 10.0 Å. Based on the size of the primary and secondary ligands, the user can adjust the `secradius` value to ensure that the secondary ligands are sampled around the longest axis of the primary ligand. The center of the spherical sampling region for the secondary ligands is the same as the geometric center of the primary ligand.

2. **Evaluate docked poses:**

   The SILCS-MC tool set allows users to easily evaluate docked poses generated from the SILCS-MC docking runs. Users can rank order and collect the docked poses using `2_calc_lgfe_min_avg_sd` (see *Best-Pose Retrieval*) and extract SILCS simulation snapshots in which the protein conformation best complements the docked conformation and orientation of the ligand using `3_scan_traj` (see *Best Protein–Ligand Complex Retrieval*).

   For details on evaluating the resulting SILCS-MC docked poses, please refer to *Assessment of SILCS-MC Docked/Refined Poses in CLI*.

SILCS-MC Docking Protocol Details

On computers using NVIDIA GPU with CUDA version 11 or later, SILCS-MC Docking spawns one independent job on one GPU and one CPU per ligand. On computers without NVIDIA GPU, SILCS-MC Docking spawns five independent single-core serial jobs per ligand. Each SILCS-MC run involves 1250 cycles (on computers with NVIDIA GPU) or a maximum of 250 cycles and a minimum of 50 cycles (on computers without NVIDIA GPU) of MC/SA sampling of the ligand within a defined spherical sampling space. Each of these cycles, regardless of GPU or CPU usage, consists of 10,000 steps of MC at a high temperature followed by 40,000 steps of SA towards a lower temperature. At the beginning of each cycle, the ligand will be reoriented within the predefined sphere. The MC sampling has three types of moves: i) molecular translations with a maximum step size of 1 Å, ii) molecular rotation with a maximum step size of 180 degrees, and iii) intramolecular dihedral rotations with a maximum step size of 180 degrees. For intramolecular dihedral rotations, only the rotatable dihedral angles are selected for MC moves. The lowest LGFE scoring pose from the MC sampling is used as starting pose in the following SA sampling. The SA sampling also involves the same three types of moves, but with a smaller step size compared to the MC sampling: i) molecular translations with a maximum step size of 0.2 Å, ii) molecular rotation with a maximum step size of 9 degrees and, iii) intramolecular dihedral rotations with a maximum step size of 9 degrees. The lowest LGFE scoring pose from the SA is saved in a multi-frame SD file: `3_silcsmc/<run>/out/<lig>.sdf`.

On computers using NVIDIA GPU, the MC/SA procedure continues until 1250 cycles have been completed. On computers without GPU, the LGFE score difference between the top three poses (defined by lowest LGFE scores) are evaluated after a minimum of 50 MC/SA cycles. If the LGFE score difference between the 3 poses is less than 0.5 kcal/mol, then that run is considered converged

and terminated. If the top three scored poses are separated by more 0.5 than kcal/mol, the MC/SA procedure continues either until the convergence criterion is met or until a maximum of 250 MC/SA cycles have been completed with the most favorable LGFE pose selected.

SILCS-MC Docking with Secondary Ligands (Water Molecules)

When including secondary ligands, the SILCS-MC docking protocol extends the MC steps to include the secondary ligands in addition to the primary ligand. The MC step sizes for the secondary ligands (by default water molecules) are set to 0.5 Å for translation, 180 degrees for rotation, and 180 degrees for dihedral rotation. The SA step sizes for the secondary ligands are set to 0.2 Å for translation, 9 degrees for rotation, and 9 degrees for dihedral rotation. The sampling region for the secondary ligands is set to a sphere centered at the geometric center of the primary ligand with a 10 Å (default) radius. The inclusion of water molecular as secondary ligands in SILCS-MC docking has been validated against crystal structures of protein–ligand complexes containing water mediated interactions [19].

6.5.3 SILCS-MC Pose Refinement

The pose refinement protocol is designed to limit conformational sampling near the ligand input pose supplied by the user. Pose refinement is appropriate when there is high confidence in the input parent ligand pose. The sphere center for the pocket definition is automatically and independently computed for each and every ligand and is the center-of-geometry of that ligand's input coordinates.

SILCS-MC Pose Refinement Using the SilcsBio GUI

To use SILCS-MC Pose Refinement in the SilcsBio GUI, please see the previous section on running SILCS-MC docking from the SilcsBio GUI (*SILCS-MC Docking Using the SilcsBio GUI*); in Step 3, select the "Pose Refinement" option. Note that there will be no "Select Pocket" step, as pose refinement assigns the pocket center based on the center-of-geometry of the input ligand on a per-ligand basis. For multiple input ligands in a single `.sdf` file, each ligand will have its own center-of-geometry used to define the pocket center.

SILCS-MC Pose Refinement Using the CLI

1. **Launch SILCS-MC pose refinement runs:**

   To set up and run SILCS-MC pose refinement, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate `.mol2` or `.sdf` file. Alternatively, all the ligands can be combined into a single `.sdf` file. With this information, enter the following command to set up and launch SILCS-MC refinement runs:

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_local prot=<prot pdb> ligdir=
↪<directory containing ligand mol2/sdf>
```

Required parameters:

- Path and name of protein PDB file:

```
prot=<protein pdb file>
```

- Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<location and name of directory containing ligand mol2/
↪sdf>
```

If the ligdir option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

> **Warning:** Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

Optional parameters:

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

- Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will overwrite␣
↪ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

- Alignment to reference molecule prior to SILCS-MC docking:

```
refsdf=<location and name of SD file of reference molecule to␣
↪align to; default=None>
```

If the `refsdf` option is used, the ligands in `sdfile` will be aligned to the reference molecule before the SILCS-MC docking runs are initiated. The alignment is performed using the Open3DAlign(O3A) algorithm [17] available in RDKit. Currently, this option is only available using the `sdfile` option, and is not available using the `ligdir` option.

- Inclusion of halogen FragMaps:

```
halogen=<use halogen maps; true/false; default=false>
```

If halogen FragMaps have been generated (see *Setup with halogen probes*), they can be included in the SILCS-MC calculation to improve the scoring of halogen-containing compounds. To do so, add the `halogen=true` option.

- Atom classification scheme (ACS):

```
class=<atom classification scheme (generic or specific);␣
↪default=generic>
```

Atomic GFEs are assigned based on the atom's classification and its overlap with the corresponding FragMap. The atom's classification is assigned based on the ACS used. Users can choose between generic and specific ACS using the `class` option. The default ACS is the `generic` ACS in which the generic apolar, donor, and acceptor maps are used to score corresponding atoms. In the `specific` ACS, specific maps (e.g., formamide nitrogen FragMaps and imidizole donor nitrogen FragMaps are kept separate and are not grouped into generic donor FragMaps) to score corresponding atoms. The schemes are stored in `${SILCSBIODIR}/data/silcs/`. Further details regarding ACS and how they affect atomic GFEs and LGFEs are provided in ref. [18].

- Option to bundle jobs:

```
bundle=<bundle multiple (single) jobs into larger jobs; true/
↪false; default=false>
```

When `bundle=true` multiple (single) jobs will be bundled into a single larger job. The number of jobs to be bundled can be set with the `nproc` parameter.

- Number of jobs to bundle (when `bundle=true`):

```
nproc=<number of jobs to bundle; default=8>
```

By default number of jobs to bundle is determined by the number of CPU cores on the compute nodes available to the user. The user can override this by setting the `nproc` parameter.

- Path to python executable:

```
python=path to python executable; default=$(which python)>
```

The default python path will be checked using the `which python` command.

- Use GPU version of SILCS-MC:

```
gpu=<use GPU; true/false; default=true>
```

By default, the GPU version of SILCS-MC is used. To use the CPU version, use `gpu=false`. The GPU version of SILCS-MC is only available on computers with NVIDIA GPUs and CUDA version 11 or later. The CPU version of SILCS-MC is available on all machines, however it is significantly slower than the GPU version. When using the CPU version of SILCS-MC, the SILCS-MC calculation will be split into five independent, single-core serial jobs per ligand. Also note that the `nproc` parameter must not be changed from its default value when using the `gpu=false bundle=true` options to bundle CPU jobs.

Secondary Ligand Parameters:

- Perform SILCS-MC Pose Refinement with secondary ligands:

```
seclig=<true/false; default=false>
```

By default, the SILCS-MC pose refinement will be performed with the primary ligand only. Users can perform SILCS-MC pose refinement on the primary ligand with SILCS-MC docking on multiple ligands simultaneously by setting `seclig=true`. When set to true, SILCS-MC pose refinment will be performed on the primary ligand (specified by `ligdir` or `sdfile`) and SILCS-MC docking will be performed on the secondary ligand. By default, 15 water molecules will be used as secondary ligands, however, the secondary ligand may also be specified with `secligdir` or `secsdfile` (described below).

- Path and name of directory containing secondary ligand SDF or Mol2 files:

```
secligdir=<path to directory containing secondary ligand mol2/
↪sdf files>
```

When `seclig=true`, by default the `secligdir` option is internally set to `${SILCSBIODIR}/templates/silcs-mc/secligdir`, which contains 15 water molecules. Users can provide a different set of secondary ligands by setting the `secligdir` option to the path of the directory containing the desired secondary ligand Mol2 or SD files.

Tip: To increase or decrease the number of water molecules simultaneously docked with the primary ligand, the user can copy the desired number of water molecules from the default `secligdir` directory (`${SILCSBIODIR}/templates/silcs-mc/`

`secligdir`) to a new directory and set the `secligdir` option to the new directory. It does not matter if the coordinates of two water molecules are the same, as long as the file names are different.

- Path and name of secondary ligand SD file:

```
secsdfile=<path to sdf file of the secondary ligands; this␣
↪option will overwrite secligdir>
```

If the `secsdfile` option is used, the `secligdir` option is not needed, and any input for the `secligdir` option will be overwritten. The `secsdfile` option is recommended if the user has an SDF file containing all secondary ligands to be used.

2. **Evaluate refined poses:**

   The SILCS-MC tool set allows users to easily evaluate refined poses generated from the SILCS-MC refinement runs. Users can rank order and collect the resulting poses using `2_calc_lgfe_min_avg_sd` (see *Best-Pose Retrieval*) and extract SILCS simulation snapshots in which the protein conformation best complements the refined conformation and orientation of the ligand using `3_scan_traj` (see *Best Protein–Ligand Complex Retrieval*).

   For details on evaluating the resulting SILCS-MC refined poses, please refer to *Assessment of SILCS-MC Docked/Refined Poses in CLI*.

SILCS-MC Pose Refinement Protocol Details

On computers using NVIDIA GPU with CUDA version 11 or later, SILCS-MC Pose Refinement spawns one independent job on one GPU and one CPU per ligand. On computers without NVIDIA GPU, SILCS-MC Pose Refinement spawns one independent single-core serial jobs per ligand. Each SILCS-MC run involves 1250 cycles (on computers with NVIDIA GPU) or 50 cycles (on computers without NVIDIA GPU) of MC/SA sampling of the ligand within a 10 Å sphere. The center of the sphere is defined as the center-of-geometry of the input ligand pose. Each of these cycles, regardless of GPU or CPU usage, starts with the ligand in the input pose and then consists of 100 steps of MC at high temperature followed by 1000 steps of SA towards a lower temperature. At the beginning of each cycle, the ligand orientation/conformation will be reset to the one found in the input file. MC sampling moves are: i) molecular translation with a maximum step size of 0.5 Å, ii) molecular rotation with a maximum step size of 15 degrees, and iii) intramolecular dihedral rotation with a maximum step size of 180 degrees. For intramolecular dihedral rotation, only the rotatable dihedral angles are selected for MC moves. The lowest LGFE scoring pose from the MC sampling is used as starting pose in the following SA sampling. SA sampling moves are smaller than for the MC phase: i) molecular translation with a maximum step size of 0.2 Å, ii) molecular rotation with a maximum step size of 9 degrees, iii) intramolecular dihedral rotation with a maximum step size of 9 degrees. The lowest LGFE scoring pose from the SA is saved in the multi-frame SD file `3_silcsmc/<run>/out/<lig>.sdf`.

On computers using NVIDIA GPU, the MC/SA procedure continues until 1250 cycles have been completed. On computers without GPU, the LGFE score difference between the top three poses (defined by lowest LGFE scores) are evaluated after a minimum of 50 MC/SA cycles. If the LGFE score difference between the 3 poses is less than 0.5 kcal/mol, then that run is considered converged and terminated. If the top three scored poses are separated by more 0.5 than kcal/mol, the MC/SA procedure continues either until the convergence criterion is met or until a maximum of 250 MC/SA cycles have been completed with the most favorable LGFE pose selected.

For secondary ligands, the protocol is same as in the SILCS-MC docking protocol as described in *SILCS-MC Docking with Secondary Ligands (Water Molecules)*.

6.5.4 SILCS-MC Pose Evaluation

SILCS-MC pose evaluation allows the user to evaluate the LGFE of a crystal ligand or a ligand already docked to a target protein for which FragMaps are available. The SILCS-MC pose evaluation protocol evaluates the LGFE score based on the input pose of the ligand. No conformational sampling of the ligand is performed.

SILCS-MC Pose Evaluation Using the SilcsBio GUI

To calculate the LGFE of a pose without re-docking or refining the ligand conformation and orientation, please see the previous section on running SILCS-MC docking from the SilcsBio GUI (*SILCS-MC Docking Using the SilcsBio GUI*); in Step 3, select the "Pose Evaluation" option. Note that there will be no "Select Pocket" step, as pose evaluation will only calculate the LGFE of the input ligand in its original orientation and conformation.

SILCS-MC Pose Evaluation Using the CLI

1. **Launch SILCS-MC pose evaluation runs:**

   To set up and run SILCS-MC pose evaluation, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate `.mol2` or `.sdf` file. Alternatively, all the ligands can be combined into a single `.sdf` file. With this information, enter the following command to set up and launch SILCS-MC refinement runs:

   ```
   ${SILCSBIODIR}/silcs-mc/1_run_silcsmc_fixedpose prot=<prot pdb>␣
   ↪ligdir=<directory containing ligand mol2/sdf>
   ```

   Required parameters:

   - Path and name of protein PDB file:

     ```
     prot=<protein pdb file>
     ```

- Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<location and name of directory containing ligand mol2/
↪sdf>
```

If the ligdir option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

> **Warning:** Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

Optional parameters:

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

- Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will overwrite␣
↪ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed, and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

- Inclusion of halogen FragMaps:

```
halogen=<use halogen maps; true/false; default=false>
```

If halogen FragMaps have been generated (see *Setup with halogen probes*), they can be included in the SILCS-MC calculation to improve the scoring of halogen-containing compounds. To do so, add the `halogen=true` option to the `1_run_silcsmc_exhaustive` command.

- Atom classification scheme (ACS):

```
class=<atom classification scheme (generic or specific);␣
↪default=generic>
```

Atomic GFEs are assigned based on the atom's classification and its overlap with the corresponding FragMap. The atom's classification is assigned based on the ACS used. Users can choose between generic and specific ACS using the `class` option. The default ACS is the `generic` ACS in which the generic apolar, donor, and acceptor maps are used to score corresponding atoms. In the `specific` ACS, specific maps (e.g., formamide nitrogen FragMaps and imidizole donor nitrogen FragMaps are kept separate and are not grouped into generic donor FragMaps) to score corresponding atoms. The schemes are stored in `${SILCSBIODIR}/data/silcs/`. Further details regarding ACS and how they affect atomic GFEs and LGFEs are provided in ref. [18].

- Option to bundle jobs:

```
bundle=<bundle multiple (single) jobs into larger jobs; true/
↪false; default=false>
```

When `bundle=true` multiple (single) jobs will be bundled into a single, larger job. The number of jobs to be bundled can be set with the `nproc` parameter.

- Number of jobs to bundle (when `bundle=true`):

```
nproc=<number of jobs to bundle; default=8>
```

- Path to python executable:

```
python=path to python executable; default=$(which python)>
```

The default python path will be checked using the `which python` command.

- Use GPU version of SILCS-MC:

```
gpu=<use GPU; true/false; default=true>
```

By default, the GPU version of SILCS-MC is used. To use the CPU version, set `gpu=false`. The GPU version of SILCS-MC is only available on computers with NVIDIA GPUs and CUDA version 11 or later. The CPU version of SILCS-MC is available on all machines, however it is significantly slower than the GPU version. When using the CPU version of SILCS-MC, the SILCS-MC calculation will be performed as a single independent, single-core serial job per ligand. Also note that the `nproc` parameter must not be changed from its default value when using the `gpu=false bundle=true` options to bundle CPU jobs.

Secondary Ligand Parameters:

- Perform SILCS-MC Pose Evaluation with secondary ligands:

```
seclig=<true/false; default=false>
```

By default, the SILCS-MC pose evaluation will be performed on the primary ligand only. Users can perform SILCS-MC pose evaluation on multiple ligands simultaneously by setting `seclig=true`. When set to true, the user will have to provide a set of secondary ligands with either the `secligdir` or `secsdfile` options as described below.

- Path and name of directory containing secondary ligand SDF or Mol2 files:

```
secligdir=<path to directory containing secondary ligand mol2/
↪sdf files>
```

Users can provide a set of secondary ligands by setting the `secligdir` option to the path of the directory containing the secondary ligand Mol2 or SD files.

- Path and name of secondary ligand SD file:

```
secsdfile=<path to sdf file of the secondary ligands; this␣
↪option will overwrite secligdir>
```

If the `secsdfile` option is used, the `secligdir` option is not needed, and any input for the `secligdir` option will be overwritten. The `secsdfile` option is recommended if the user has an SDF file containing all secondary ligands to be used.

2. **Evaluate poses:**

   To extract the LGFE scores calculated by SILCS-MC pose evaluation, use `2_calc_lgfe_min_avg_sd` following the instructions detailed in *Best-Pose Retrieval*. Users can additionally extract SILCS simulation snapshots in which the protein conformation best complements the refined conformation and orientation of the ligand using `3_scan_traj` (see *Best Protein–Ligand Complex Retrieval*).

SILCS-MC Pose Evaluation Protocol Details

Pose evaluation initiates one single-GPU and single-CPU job (or single-core serial job on computers without NVIDIA GPU or without CUDA version 11 or later) per ligand, and involves a single step evaluation of LGFE. Technically, all step sizes are set to zero and only 1 step of MC and SA is performed in order to get the output LGFE. When including secondary ligands, all step sizes for the secondary ligands are also set to zero.

6.5.5 SILCS-MC Docking in Restrained Mode

SILCS-MC restrained docking allows the user to dock ligands that form covalent bonds with the target protein. SILCS-MC restrained docking involves exhaustive sampling of a ligand's conformation in a given pocket to determine its most favorable orientation, as in SILCS-MC docking with the exception that the ligand sampling is restricted by a constraint on a user-defined reactive atom of the ligand. All other sampling parameters are the same as in SILCS-MC docking.

SILCS-MC Restrained Mode Docking Using the CLI

1. **Launch SILCS-MC Restrained Mode docking runs:**

   To set up and run SILCS-MC Restrained Mode docking, create a directory containing all of the ligands to be evaluated. Each ligand can be stored as a separate Mol2 or SD file. Alternatively, all the ligands can be combined into a single SD file. In addition, a restraints file that specifies the atom index number of the atom to be restrained for each ligand is required. With this information, enter the following command to set up and launch SILCS-MC docking runs in Restrained Mode:

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_restrained prot=<prot pdb> \
   ligdir=<directory containing ligand mol2/sdf> \
   rstrfile=<location and name of restraints file>
```

   Required parameters:

   - Path and name of protein PDB file:

```
prot=<protein pdb file>
```

   - Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<location and name of directory containing ligand mol2/
↪sdf>
```

     If the ligdir option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

> **Warning:** Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

Tip: If using the SILCS-MC restrained mode docking to approximate the binding of

covalently bonded inhibitors, then the chemical structure of the ligand in the covalent bond should be considered when preparing the ligand Mol2/SD file. The ligands being docked can be prepared with the reactive atom in the form after a covalent bond has been formed with the target protein. Hydrogen atoms could be temporarily added to the reactive atom (and adjacent atoms as needed) to facilitate the restrained docking. For example, the C=C double bond of an acrylamide molecule could be changed to a C-C single bond for a covalently bonded acrylamide moiety, where the constraint imposed on the ligand C atom involved in covalent bond being formed with, for example, the S of a Cysteine. Note that the ligand's chemical structure must be a stable species.

- Path and name of restraints file:

```
rstrfile=<location and name of restraints file>
```

The restraints file informs the program which atom(s) in the ligand are restrained in the docking runs. The restraints file should contain the following information:

`<ligand_name> <atom_index> <x> <y> <z> <radius>`

where,

  - `ligand_name`: name of the ligand in the Mol2/SD file (not the file name)
  - `atom_index`: index of the atom to be constrained
  - `x, y, z`: coordinates for the center of the constraint in Angstroms
  - `radius`: radius of the constraint in Angstroms (Recommended: 0.5 - 1.5)

Optional parameters:

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

- Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file; this option will overwrite␣
↪ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

- Alignment to reference molecule prior to SILCS-MC docking:

```
refsdf=<location and name of SD file of reference molecule to␣
↪align to; default=None>
```

If the `refsdf` option is used, the ligands in `sdfile` will be aligned to the reference molecule before the SILCS-MC docking runs are initiated. The alignment is performed using the Open3DAlign(O3A) algorithm [17] available in RDKit. Currently, this option is only available using the `sdfile` option, and is not available using the `ligdir` option.

- Inclusion of halogen FragMaps:

```
halogen=<use halogen maps; true/false; default=false>
```

If halogen FragMaps have been generated (see *Setup with halogen probes*), they can be included in the SILCS-MC calculation to improve the scoring of halogen-containing compounds. To do so, add the `halogen=true` option to the `1_run_silcsmc_exhaustive` command.

- Atom classification scheme:

```
class=<atom classification scheme (generic or specific);␣
↪default=generic>
```

Users can use custom atom classification schemes if they wish.

- Option to bundle jobs:

```
bundle=<bundle multiple (single) jobs into larger jobs; true/
↪false; default=false>
```

When `bundle=true` multiple (single) jobs will be bundled into a single larger job. The number of jobs to be bundled can be set with the `nproc` parameter.

- Number of jobs to bundle (when `bundle=true`):

```
nproc=<number of jobs to bundle; default=8>
```

- Path to python executable:

```
python=path to python executable; default=$(which python)>
```

The default python path will be checked using the `which python` command.

Additional Parameters

- Center of the spherical sampling region:

```
center=<"x,y,z"; e.g., center="20,30,-2"; default=ligand's␣
↪center of mass>
```

If the center is not specified by the user, the default center will be calculated from the input ligand's center of mass, and the initial conformation of the ligand in the SILCS-MC docking runs will correspond to the conformation of the ligand in the input Mol2 or SD file. If the center is specified, then the conformation and orientation of the ligand will be randomized prior to the SILCS-MC docking runs.

- Radius of the spherical sampling region:

```
radius=<radius from center in Å; default=10.0>
```

2. **Evaluate docked poses:**

   The SILCS-MC tool set allows users to easily evaluate docked poses generated from the SILCS-MC docking runs. Users can rank order and collect the docked poses using `2_calc_lgfe_min_avg_sd` (see *Best-Pose Retrieval*) and extract SILCS simulation snapshots in which the protein conformation best complements the docked conformation and orientation of the ligand using `3_scan_traj` (see *Best Protein–Ligand Complex Retrieval*).

   For details on evaluating the resulting SILCS-MC docked poses, please refer to *Assessment of SILCS-MC Docked/Refined Poses in CLI*.

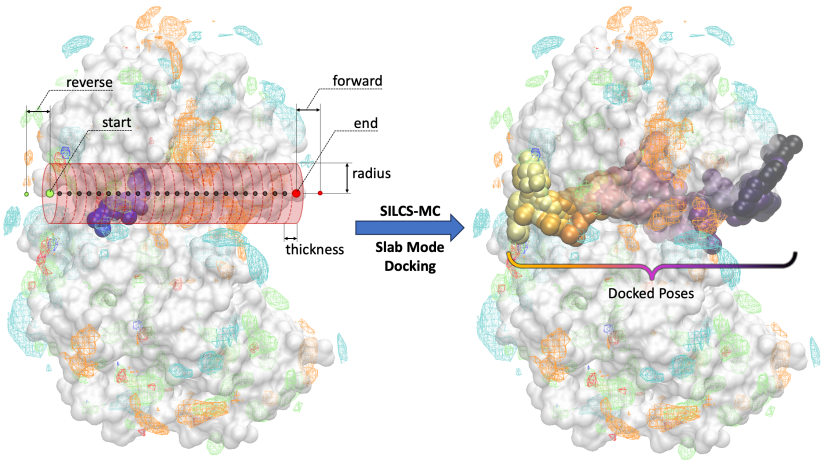
SILCS-MC Restrained Mode Docking Protocol Details

The SILCS-MC Restrained Mode docking protocol is similar to the SILCS-MC docking protocol, but with the addition of a contraint on select atoms of the ligand. The restrained atom(s) are specified by the user in the restraints file. Additionally, the restraints file should contain the center and the radius of the contraint. The radius of the constraint is the distance from the center within which the restrained atom(s) are allowed to move. A radius of 0.5 - 1.5 Å is recommended.

6.5.6 SILCS-MC Docking in Slab Mode

For a ligand to reach the active site of a protein, the ligand must diffuse into the active site. Characterizing ligand binding pathways can provide insights into ligand binding/unbinding kinetics and reveal potential protein "gateways" into active sites, which may be targeted for the binding of small molecules to block the active site. The Slab Mode of SILCS-MC docking allows users to dock ligands along a user-defined pathway.

In the Slab Mode of SILCS-MC docking, the user defines the "start" and "end" point (large green and red circles in the figure below, respectively) of a linear pathway for the ligand to be docked along. With the user-defined pathway as the axis, a series of cylindrical sampling regions (red cylinders in the figure below) are constructed with a tunable "radius" and "thickness" spanning the entire length between the start and end point. Users may optionally add additional sampling cylinders beyond the start ("reverse") and end ("forward") points (small green and red circles in the figure below, respectively). After running SILCS-MC with Slab Mode, the docked poses of the ligand along the pathway, in sequential order, will be generated (right panel of the figure below).



SILCS-MC Slab Mode Docking Using the CLI

Ligands can be docked along a user-defined pathway using the SILCS-MC Slab Mode docking through the following steps:

1. **Launch SILCS-MC Slab Mode Docking:**

   To set up and run SILCS-MC docking using the Slab Mode from the command line interface, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate SDF or Mol2 file. Alternatively, all the ligands can be combined into a single SD file. Additionally, determine the start and end point, in Cartesian coordinates, of the pathway along which the ligand will be docked. With this information, enter the following command to set up and launch the Slab Mode SILCS-MC docking run:

```
 $SILCSBIODIR/silcs-mc/1_run_silcsmc_slab prot=<prot pdb> ligdir=
↪<ligand directory> \
    start=<"x,y,z"> end=<"x,y,z">

*Required parameters:*
```

   - Path and name of protein PDB file:

```
prot=<protein pdb file>
```

   - Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<ligand directory>
```

If the ligdir option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

- Starting point of the pathway along which the ligand(s) will be docked:

```
start=<"x,y,z"; e.g., start="20,30,-2">
```

- End point of the pathway along which the ligand(s) will be docked:

```
end=<"x,y,z"; e.g., end="20,30,22">
```

Optional parameters:

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

- Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

- Increments in which the ligand will be docked along the pathway (thickness of the slab):

```
thickness=<thickness/height of the slab; default=2.0 A>
```

- Sampling radius for each docking run along the pathway (radius of the slab):

```
radius=<radius of the slab; default=5.0 A>
```

- Extension of sampling beyond end point:

```
forward=<extra slab number of forward sampling after end;␣
↪default=0>
```

- Extension of sampling beyond starting point:

```
reverse=<extra slab number of reverse sampling before start;␣
↪default=0>
```

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will overwrite␣
↪ligdir>
```

If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

2. **Evaluate docked poses:**

   The SILCS-MC tool set allows users to easily evaluate docked poses generated from the SILCS-MC Slab Mode docking runs. Users can collect the docked poses along the specified pathway using `2_calc_lgfe_min_avg_sd` with the `slab=true` parameter (see *Best-Pose Retrieval*). In addition, using `2_calc_lgfe_min_avg_sd` will additionally collect and rank order the resulting docked poses. SILCS simulation snapshots in which the protein conformation best complements the refined conformation and orientation of the ligand using can also be extracted with `3_scan_traj` (see *Best Protein–Ligand Complex Retrieval*).

   For details on evaluating the resulting SILCS-MC Slab Mode docked poses, please refer to *Assessment of SILCS-MC Docked/Refined Poses in CLI*.

SILCS-MC Slab Mode Docking Protocol Details

SILCS-MC Slab Mode docking spawns a single gpu job per ligand. In SILCS-MC Slab Mode docking, the pathway along which the ligand(s) will be docked is divided into cylindrical slabs. By default, the cylindrical slabs are 2 Å thick with a radius of 5 Å. The thickness and radius of the cylindrical slabs can be customized in the command line interface (`thickness` and `radius` optional parameters; see above). For each cylindrical slab, a maximum of 250 cycles and a minimum of 50 cycles of Monte Carlo/Simulated Annealing (MC/SA) sampling of the ligand within the cylindrical slab is performed. Each of these 250 cycles consists of 10,000 steps of MC at a high temperature followed by 40,000 steps of SA towards a lower temperature. At the begining of each cycle, the ligand will be reoriented within the cylindrical slab. The MC sampling has three types of moves: i) molecular translations with a maximum step size of 1.0 Å, ii) molecular rotations with a maximum step size of 180.0°, and iii) intramolecular dihedral rotations with a maximum step size of 180.0°. For intramolecular dihedral rotations, only the rotatable dihedral angles are selected for MC moves. The lowest LGFE scoring pose from the MC sampling is used as a starting pose in the following SA sampling. The SA sampling also involves the same three types of moves, but with a smaller step size compared to the MC sampling: i) molecular translations with a maximum step size of 0.2 Å, ii) molecular rotations with a maximum step size of 9° and, iii) intramolecular dihedral rotations with a maximum step size of 9°. For each cylindrical slab, after a minimum of 50 MC/SA cycles, if the LGFE score difference between the top three poses (defined by lowest LGFE scores) are less than 0.5 kcal/mol, then the run is considered converged and terminated. If the top three scored poses are separated by more 0.5 than kcal/mol, then the MC/SA procedure continues either until the convergence criterion is met or until a maximum of 250 MC/SA cycles have been completed. At the end of all cycles per cylindrical slab, the lowest LGFE scoring pose for each cylindrical slab is saved and stored in a multi-frame SD file (3_silcsmc/1/out/<lig>.sdf)

with each frame corresponding to the lowest LGFE scoring pose of a given cylindrical slab along the user-defined pathway.

6.5.7 Assessment of SILCS-MC Docked/Refined Poses in CLI

Upon completion of the SILCS-MC docking and/or refinement simulations, the resulting docked poses can be assessed using tools available with your SilcsBio software. To extract the structures of docked poses and evaluate their LGFE scores (energetic favorability) and 4DBA scores (bioavailability), follow the instructions in *Best-Pose Retrieval*. To extract protein conformations that complement SILCS-MC docked structures and generate a protein–ligand complex structure, follow the instructions in *Best Protein–Ligand Complex Retrieval* after extracting the docked poses through *Best-Pose Retrieval*.

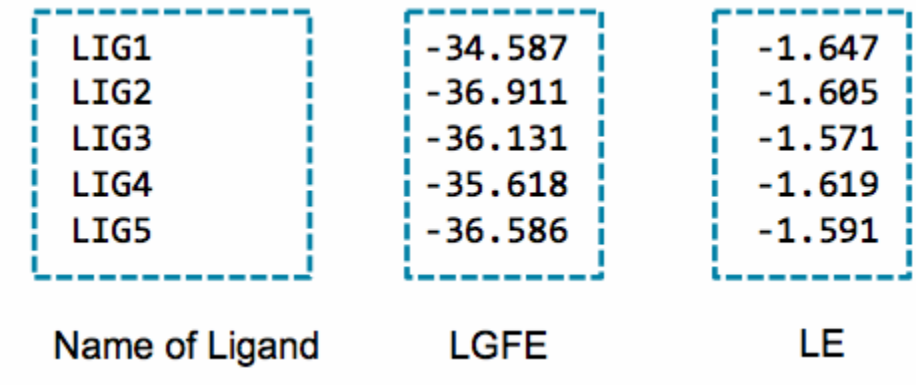
Best-Pose Retrieval

Once the SILCS-MC simulation is finished, users can retrieve the LGFE scores for each ligand subjected to SILCS-MC using the following command:

```
$SILCSBIODIR/silcs-mc/2_calc_lgfe_min_avg_sd ligdir=<directory containing␣
↪ligand mol2/sdf>
```

The above command will rank order and collect the most energetically favorable (lowest LGFE) docked poses resulting from the SILCS-MC docking runs. By default, the corresponding LGFE scores per ligand will be output in the `<silcsmcdir>/lgfe.csv` file (see below). The most energetically favorable binding pose(s) will be stored in the `<silcsmcdir>/minconfpdb/` folder in SDF file format.

Below is an example of the contents of the `lgfe.csv` output from this script:



An alternative to the LGFE score is the ligand efficiency (LE). The LE is calculated as the LGFE score divided by the number of heavy atoms in each ligand.

$$LE = \frac{LGFE}{N_{\text{HeavyAtoms}}}$$

Required parameter:

- Path and name of directory containing ligand Mol2/SDF files:

```
ligdir=<location and name of directory containing ligand mol2/sdf>
```

The same directory containing the ligand Mol2/SDF files specified with `ligdir` in the previous `1_run_silcsmc_*` step should be used if the poses for all docked ligands are being collected. If only the poses for a subset of the docked ligands is desired, then a new directory containing the Mol2/SDF files of the subset of ligands can be specified. If `sdfile` was used in the previous step, use the same `sdfile` parameter for this step.

Optional parameters:

- Name of the directory containing the SILCS-MC output:

```
silcsmcdir=<name of output directory; default=3_silcsmc>
```

The `silcsmcdir` should correspond to the `silcsmcdir` specified in the previous `1_run_silcsmc_exhaustive` step. If no `silcsmcdir` was specified in the previous step, then the default `silcsmcdir=3_silcsmc` should be used.

- Path and name of ligand SD file:

```
sdfile=<location and name of SD file, this option will overwrite␣
↪ligdir>
```

The same SD file should be used as in the previous `1_run_silcsmc_exhaustive` step. If the `sdfile` option is used, the `ligdir` option is not needed and any input for the `ligdir` option will be overwritten. The `sdfile` option is recommended if the user has an SD file containing all ligands under investigation.

- Number of output poses:

```
npose=<number of best scoring poses to collect; default=1>
```

By default, only the best scoring, based on LGFE, docked pose will be output. To ouput additional docked poses, use the `npose` parameter and specify the desired number of docked poses. E.g., if 5 poses are desired, using `npose=5` will result in the 5 best scoring poses being output.

- Collect diverse poses based on RMSD clustering:

```
rmsd_cluster=<when npose more than 1, cluster by RMSD; true/false;␣
↪default=true>
```

When `rmsd_cluster=true` and `npose` is greater than 1, the next best scoring poses will be selected if it is not within the RMSD cutoff of the any of the previously selected poses. The

RMSD cutoff is set to 3.0 Å by default and <rmsd_cluster_id> is already calculated during the SILCS-MC docking. Here we are just processing that information to collect diverse poses.

- Option to output docked poses in PDB format:

```
pdb=<write PDB file in minconfpdb folder; true/false; default=false>
```

When `pdb=true` the docked poses will be output in PDB format in addition to the default SDF file format.

- Option to output ligand SMILES strings:

```
smiles=<write SMILES string to lgfe.csv; true/false; default=false>
```

When `smiles=true` the output `lgfe.csv` file will include the ligand SMILES strings in the second column of the file.

- Option to output RMSD w.r.t input pose:

```
ligrmsd=<write ligand RMSD w.r.t. input pose to lgfe.csv; true/false;
↪ default=false>
```

When `ligrmsd=true` the output `lgfe.csv` file will include a LigRMSD column and LigCOMD column. The LigRMSD column will be populated with the RMSD of the docked pose with respect to the input pose. The LigCOMD will be populated with the distance between the center of geometry of the docked pose and the input pose. The input pose is the ligand conformation and orientation in the input Mol2/SDF file. Note that this RMSD is calculated during the SILCS-MC docking run and is simply printed into the output file with no additional calculations performed; providing a different input pose in this step will not change the LigRMSD or LigCOMD values.

- Option to calculate 4DBA descriptors and 4DBA-LGFE scores:

```
fourdba=<true|false; default=false>
```

When `fourdba=true` the four 4DBA descriptors of each input ligand will be calculated and output into `lgfe-4dba.csv`. A high 4DBA score indicates that the ligand is bioavailable while a low 4DBA score indicates that the ligand is not bioavailable. For more information of 4DBA and LGFE-4DBA scores, please see *4D Bioavailability (4DBA) Calculation*.

- Path to python executable:

```
python=<path to python executable for 4DBA calculations>
```

The default python path can be checked using the `which python` command. The `rdkit`, `tqdm`, `ipython` and `scipy` packages must be installed for the 4DBA score calculation. Please refer to *Python 3 Requirement* for more information.

- Option to extract LGFEs and poses along a pathway for Slab Mode (when `1_run_silcsmc_slab` was used to dock):

```
slab=<true|false; default=false>
```

When `slab=true`, an additional output file `lgfe_slab.csv` will be produced. This file will contain the ligand name, step along the pathway (slab), and the ligand LGFE, ligand efficiency (LE), and center-of-geometry at that step. In conjunction with `smiles=true`, the ligand SMILES strings will also be output into the second column of the `lgfe_slab.csv` file. In addition, an SD file containing the docked poses of the ligand along the user-defined pathway will be stored for each ligand in `3_silcsmc/conf_slab/`.

- Option to sort the output by LGFE:

```
sortbylgfe=<sort lgfe.csv output by LGFE; true/false; default=false>
```

When `sortbylgfe=true`, the output `lgfe.csv` file will be sorted by LGFE in ascending order. By default, the output `lgfe.csv` file is sorted by the ligand name in alphanumeric order when *ligdir* is used and by the ligand index in the SD file when *sdfile* is used.

- Submit job to queuing system:

```
batch=<submit job to queuing system; true/false; default=false>
```

Secondary Ligand Parameters:

If secondary ligands were used in the SILCS-MC docking, a selection criteria can be applied to the secondary ligands to determine which ligands to include in the output `lgfe.csv` file. The default selection criteria was defined for water molecules as secondary ligands and is adapted from ref. [19], in which SILCS-MC docking with waters as secondary ligands was performed and validated against crystal structures of protein–ligand complexes containing water mediated interactions.

- Distance cutoff (in Å) for secondary ligand selection:

```
distcutoff=<cutoff for max distance to select seclig in Angstroms;␣
↪default=4.0>
```

The `distcutoff` parameter is the maximum distance allowed between the ligand and secondary ligand. By default, the `distcutoff` is set to 4.0 Å.

- LGFE cutoff for secondary ligand selection:

```
lgfecutoff=<cutoff for max allowed LGFE of seclig (SLigLGFE);␣
↪default=0.5>
```

The `lgfecutoff` parameter is the maximum LGFE allowed for the secondary ligand. By default, the `lgfecutoff` is set to +0.5 kcal/mol.

- Total energy cutoff for secondary ligand selection:

```
totecutoff=<cutoff for max allowed TotE (IntE+SLigLGFE) of seclig;␣
↪default=-0.6>
```

The `totecutoff` parameter is the maximum total energy allowed for the secondary ligand. By default, the `totecutoff` is set to -0.6 kcal/mol.

- Interaction energy cutoff for secondary ligand selection:

```
intecutoff=<cutoff for max allowed interaction energy (IntE) between␣
↪lig-seclig; default=-1.1>
```

The `intecutoff` parameter is the maximum interaction energy allowed between the ligand and secondary ligand. By default, the `intecutoff` is set to -1.1 kcal/mol. Note that the interaction energy is the sum of the electrostatic and Lennard-Jones interaction energies between the ligand and the secondary ligand to be selected.

- Sort the output by total LGFE:

```
sortbytotlgfe=<sort lgfe.csv output by TotLGFE; true/false;␣
↪default=false>
```

When `sortbytotlgfe=true`, the output `lgfe.csv` file will be sorted by the total LGFE in ascending order. By default, the output `lgfe.csv` file is sorted by the ligand name in alphanumeric order.

Best Protein–Ligand Complex Retrieval

After retrieving the best ligand pose from SILCS-MC, users can visualize it in the context of SILCS FragMaps. However, since FragMaps are typically visualized with the initial protein structure, the docked ligand pose may overlap with the protein structure. The best protein–ligand complex retrival tool allows users to retrieve simulation snapshots, in which the conformation of the protein (or RNA) best complements the docked ligand, from the SILCS simulations. The most complementary conformations of the protein or RNA are determined by the favorability of non-bonded interactions between the protein/RNA and the docked ligand.

To retrieve complementary conformations of the target protein/RNA, first extract the protein/RNA trajectory from SILCS simulations by re-running the `silcs/2b_gen_maps` step with the option `traj=true`:

```
${SILCSBIODIR}/silcs/2b_gen_maps prot=<prot pdb> traj=true
```

Additionally, use `oldversion=true` if v2023.1 or older versions was used to to run `2a_run_gcmd` step, i.e., the simulation trajectories were saved without hydrogen atoms.

The above command will generate `traj.xtc` and `traj.pdb` files as output, which will be used as input for the extraction of the best protein–ligand complex structure(s).

The best protein–ligand complex retrieval tool is run using the below command. The resulting output files will be `top<n>_<prot>_<lig>.csv` and `top<n>_<prot>_<lig>.pdb`, where `<n>` is the user-defined top number of frames to extract (defined with the `top` parameter), `<prot>` is the protein PDB name, and `<lig>` is the ligand name. `top<n>_<prot>_<lig>.csv` will contain the frame number from which the protein conformation originated from and, for each frame, the electrostatic interaction energy, LJ interaction energy, and the total interaction energy between the protein and the ligand at the corresponding frame. `top<n>_<prot>_<lig>.pdb` will contain the ensemble of protein–ligand complex structures.

```
${SILCSBIODIR}/silcs-mc/3_scan_traj prot=<prot pdb> ligdir=<ligand SDF/
↪Mol2 directory>
```

Required parameters:

- Path and name of protein PDB file:

  ```
  prot=<prot pdb>
  ```

- Path and name of directory containing ligand Mol2/SDF files:

  ```
  ligdir=<ligand SDF/Mol2 directory>
  ```

  Typically this `ligdir` should be pointing to the `minconfpdb` directory output from the previous `2_calc_lgfe_min_avg_sd` step with `npose=1` option, meaning that the SDF files in the ligdir are the best docked/refined poses of the ligands with one pose per sdf file.

  Alternatively, the `sdfile` option can be used to specify the location and name of the SD file containing all ligands under investigation.

Optional parameters:

- Name of the output directory:

  ```
  outputdir=<name of output directory; default=scan_traj>
  ```

- Number of top frames to extract:

  ```
  top=<number of top frames to extract; default=10>
  ```

  The simulation snapshots are ranked by the interaction energy of the docked pose with the protein conformation extracted from the simulation snapshot. By default only the Lennard-Jones component of the interaction energy is considered, this can be changed using the `sortby` parameter. The lowest interaction energy conformation is considered the most energetically favored conformation and the corresponding frame will be ranked first.

- Number of frames to skip while processing trajectory:

```
skip=<number of frames to skip; default=100>
```

- Interaction energy component(s) used to rank the frames:

```
sortby=<type of energy to sort frames by; ELEC/LJ/TOTAL; default=LJ>
```

Users can select from electrostatic (`ELEC`), Lennard-Jones (`LJ`) or total (`TOTAL`) interaction energies. By default, the simulation snapshots are ranked based on `LJ`.

- Option to ignore hydrogen atoms:

```
noh=<ignore protein hydrogens for energy calculation; true/false;␣
↪default=false>
```

Users can choose to perform the energy calculation with protein hydrogen atoms ignored. Ligand hydrogen atoms will never be ignored. When `noh=true`, this calculation will be performed in addition to the calculation with hydrogens.

- Path and name of SD file:

```
sdfile=<location and name of SD file>
```

Only one molecule in each Mol2/SD file under `ligdir` will be processed. If you have SD file with multiple molecules, use the `sdfile` option instead of the `ligdir` option.

> **Warning:** The calculation will automatically loop over all ligands in `ligdir` or `sdfile`. If there are many ligands in `ligdir` or `sdfile`, this calculation may require a long compute time, and if `oldversion=true`, many output files may be produced.

- Number of threads to use:

```
nproc=<number of threads to use; default=4>
```

- Specify calculation for v2023.x or older SilcsBio versions:

```
oldversion=<true|fase; default=false>
```

Use `oldversion=true` if the option was used to generate traj.xtc/pdb files in `silcs/2b_gen_maps`. This option should be set to `true` if SilcsBio v2023.x or older versions were used to run the SILCS simulations (`silcs/2a_run_gcmd`).

Tip: For users with access to both the SILCS-Small Molecule Suite and the CGenFF Suite, running standard MD simulation can be helpful to evaluate or refine the protein–ligand complex structure obtained from SILCS-MC Docking or Pose Refinement. The structure retrieved

from the process described in *Best Protein–Ligand Complex Retrieval* may be used to create the initial protein–ligand complex structure for further refinement using MD simulations. For more information on running standard MD simulations through the CGenFF Suite, please refer to *Standard Molecular Dynamics (MD) Simulations*.

6.5.8 User-Defined Protocols through the CLI

In addition to the default docking, pose refinement, and pose evaluation protocols, users can define their own SILCS-MC protocols. To do so, copy `${SILCSBIODIR}/templates/silcs-mc/params_custom.tmpl` to the location where you intend to run your custom SILCS-MC protocol. Edit this copy to reflect your customization; see below for a detailed description of user-definable parameters. Parameter values in angle brackets in this file, such as `<SILCSBIODIR>`, will be replaced automatically at runtime where possible.

After you have edited `params_custom.tmpl`, use the following command to set up and run SILCS-MC. Each input ligand can be stored as a separate SDF or Mol2 file. Alternatively, all input ligands can be combined in a single SD file.

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_custom prot=<prot pdb> \
   ligdir=<directory containing ligand sdf/mol2> \
   mapsdir=<directory containing SILCS FragMaps> \
   paramsfile=<params_custom.tmpl>
```

The number of runs that will be spawned can also modified with the command-line parameter `totruns`:

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_custom prot=<prot pdb> \
   ligdir=<directory containing ligand sdf/mol2> \
   mapsdir=<directory containing SILCS FragMaps> \
   paramsfile=<params_custom.tmpl> \
   totruns=<# of runs>
```

Note: `.sdf`, `.sd`, or `.mol2` files can be placed in the `ligdir` directory, and SILCS-MC will read a single molecule from each file. If a file contains multiple molecules, use of the `ligdir` option will result in only the first molecule in the file being processed.

If you have an SD file with multiple molecules in it, replace `ligdir=<directory containing ligand mol2/sdf>` with `sdfile=<path to sdfile>` to process all molecules in the file.

Warning: Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

The full list of user-definable parameters for `params_custom.tmpl` is:

- `CGENFF_RULES <cgenff rules_file>` (required)

  This file is needed by the internal CGenFF library to determine the correct force-field parameters for the ligand. The default value is `${SILCSBIODIR}/data/cgenff/cgenff.rules`

- `CGENFF_PAR <cgenff parameter file>` (required)

  Along with the CGENFF_RULES file, this file is needed by the internal CGenFF library to determine the correct force-field parameters for the ligand. The default value is `${SILCSBIODIR}/data/cgenff/par_all36_cgenff.prm`

- `SILCS_RULES <silcs rules file>` (required)

  This rules file is the atom classification scheme (ACS) used to map the different atoms in the ligand to the corresponding SILCS FragMap types. This mapping is used to determine the appropriate "field" that will be applied to the different atoms in the ligand when attempting an MC-move. The default value is `${SILCSBIODIR}/data/silcs/silcs_classification_rules_2021_generic_apolar_scale_1f.dat`

  When `silcs_classification_rules_2021_generic_apolar_scale_1f.dat` is used, ligand atoms are assigned using generic classifications for mapping back to the FragMaps.

  Additional rules files are available in the `${SILCSBIODIR}/data/silcs/` for other mapping schemes, including for more specific classifications and for using halogen FragMaps (see *Setup with halogen probes*).

- `GFE_CAP <default: 3.0>`

  Maximum allowable unfavorable GFE (kcal/mol) in the MC calculation.

- `RDIE <default: true>`

  When true, the distance dependent dielectric (RDIE) scheme is used to treat intramolecular electrostatics. When false, CDIE (constant dielectric scheme) is used.

- `DIELEC_CONST <default: 4>`

  Dielectric constant used in the intramolecular electrostatic interactions calculations.

- `MINIMIZE_INPUT <default: false>`

  Perform minimization of input structure.

- `MINIMIZE_BFGS <default: false>`

  Perform minimization of input structure using BFGS algorithm.

- `MIN_STEPS <default: 10000>`

  Maximum number of steps of minimization performed using the steepest-descent algorithm with the ligand, before initiating MC simulation.

- EMTOL <default: 0.01>

  Minimization is converged when the diff in total energy (totE) across the last 10 steps is smaller than this value. Once this criteria is satisfied minimization terminates.

- MC_MOVE_RANGE <default:1.0 180.0 180.0>.

  Maximum range of translation, rigid body rotation and dihedral rotation per step of MC simulation.

- MC_PRNT_FRQ <default: 0>

  Number of intermediate steps of MC to be written into OUTMCPDBFILE.

- MC_STEPS <default: 10000>

  Number of steps of MC simulation to be performed per cycle.

- MC_RUN <default: 1>

  Number of MC simulation cycles to be performed. By default, SILCS-MC performs one cycle of MC simulation with MC_STEPS. In some cases, when users want to generate poses and orientations, and then perform a local refinement for each pose, they can set MC_RUN to a higher value.

- MC_STEPS2 <default: 10000>

  Number of steps of MC simulation to be performed for later cycle of MC_RUN, only when MC_RUN is larger than 1. By default, SILCS-MC performs one cycle (MC_RUN = 1) of MC simulation with MC_STEPS, and MC_STEPS2 will be ignored.

- MC_MOVE_RANGE2 <default:0.1 90.0 60.0>.

  Maximum range of translation, rigid body rotation and dihedral rotation per step for MC_STEPS2.

- SIM_ANNEAL_MOVE_RANGE <default:0.2 9.0 9.0>

  Maximum range of translation, rigid body rotation and dihedral rotation per step of simulated annealing simulation after MC simulation.

- SIM_ANNEAL_STEPS <default: 40000>

  Number of steps of simulated annealing to be performed per cycle.

- INIT_RUNS <default: 50>

  Number of MC/SA cycles before initiating checks for convergence.

- NUM_TOL <default: 3>

  Number of top-scoring cycles with differences in LGFE less than DELTAE_TOLERANCE, before this simulation (run) is considered converged.

- DELTAE_TOLERANCE <default: 0.5>

When differences in LGFE of NUM_TOL most-favorable cycles are less than this defined tolerance value, convergence is reached and the program exits

- DELTAE_BUFF <default: 10>

  Progression of MC+SA from one cycle to next is such that LGFE (of lowest conf) from MC should be less than (prev_min+deltae_buff). This ensures that N cycles are proceeding towards a minimum lower than that previously discovered lowest energy conformation.

- TOTE_CRITERIA <default: false>

  When true, instead of LGFE, total energy (totE) of the system is used for convergence checks. Useful when running vacuum-phase MC simulations of the ligand.

- TOT_RUNS <default: 250>

  Maximum number of MC simulation cycles. The program terminates if the DELTAE_TOLERANCE criteria is satisfied before reaching TOT_RUNS. Alternately, even if the DELTAE_TOLERANCE criteria is not satisfied when the number of cycles executed reaches TOT_RUNS, the program terminates. The GPU version of SILCS-MC ignores DELTAE_TOLERANCE and runs for all TOT_RUNS cycles specified.

- RANDOM_SEED: <default: system-time>

  Seed used in MC simulation. When not set, system-time is used as a seed.

- SIMULATION_CENTER: <x,y,z>

  Cartesian coordinates of where the MC simulation should be performed.

- SIMULATION_RADIUS: <default: 10.0 A>

  Radius of the sphere within which MC simulation will be performed.

- RANDOM_INIT_ORIENT: <true/false>

  When set to TRUE, SIMULATION_CENTER should also be set. The ligand is placed within a sphere of size SIMULATION_RADIUS in a random orientation and a random conformation.

  When set to FALSE, the center-of-geometry of the ligand is used as the center for the MC simulation, and the input pose of the ligand is used as the starting pose for every MC run. This is useful when the ligand pose in the pocket is well known.

- ATOM_TO_RESTRAIN: <atom number in sdf/mol2>

  When set, a spherical potential is applied to restrain the defined atom within the sphere during MC moves. This enables geometrically restraining a particular pharmacophore feature. Note, when using this feature, supply the full molecule with explicit hydrogens already added. Rrandom pocket pose and placement using RANDOM_INIT_ORIENT true is incompatible with this option.

  When not set, the entire molecule is free to rotate/move/translate

- ATOM_RESTRAINT_CENTER: <x,y,z>

To be used in conjunction with ATOM_TO_RESTRAIN option. This value is used to defined the center of the spherical potential.

- ATOM_RESTRAINT_RADIUS: <default: 1.0 A>

  To be used in conjunction with ATOM_TO_RESTRAIN option. This value is used to defined the radius of the spherical potential. When not defined, then a default of 1 A is used.

- OUTRMSDFILE <output RMSD file>

  This file stores the RMSD and LGFE of the lowest energy conformation from each run of the MC/SA simulation. To be used in conjunction with RANDOM_INIT_ORIENT set to true.

- SILCSMAP <MapType> <map name> <scaling factor> (required)

  Multiple SILCSMAP entries are typically defined, with each entry pointing to a FragMap filename <map name>. <scaling factor> is used to scale atomic Grid Free Energies (GFEs) for <MapType> atoms in the ligand being scored. <MapType> entries here must correspond to those defined in the SILCS_RULES rules file described above.

- OUTPUT_FORMAT [SDF|PDB] (required)

  Choice of output format. SDF or PDB is supported.

- OUTPUT_FILE <output file name> (required)

  This file stores the lowest energy conformation from each cycle of the MC/SA simulation.

- LOGFILE <output log file>

  This file stores the energy statistics of the lowest energy conformation from each cycle of the MC/SA simulation.

When docking secondary ligands, use the parameters below:

- MC_MOVE_RANGE_SECLIG <default: 1.0 180.0 180.0>

  Maximum range of translation, rigid body rotation and dihedral rotation per step of MC simulation for secondary ligands.

- SIM_ANNEAL_MOVE_RANGE_SECLIG <default:0.2 9.0 9.0>

  Maximum range of translation, rigid body rotation and dihedral rotation per step of simulated annealing simulation after MC simulation for secondary ligands.

- RANDOM_INIT_ORIENT_SECLIG <true/false>

  When set to true, the secondary ligand is placed within a sphere in a random orientation. The radius and center of the sphere is defined by SIMULATION_RADIUS_SECLIG and SIMULATION_CENTER_SECLIG, respectively.

  When set to false, the input pose of the secondary ligand is used as the starting pose for every MC run.

- `SIMULATION_CENTER_SECLIG: <x,y,z>`

  Cartesian coordinates of spherical sampling regions where the MC simulation should be performed for secondary ligands. This is used when `RANDOM_INIT_ORIENT_SECLIG` is set to `true`.

- `SIMULATION_RADIUS_SECLIG: <default: 10.0 A>`

  Radius of the sphere within which MC simulations will be performed for secondary ligands. This is used when `RANDOM_INIT_ORIENT_SECLIG` is set to `true`. Users are recommended to set this value to a different value to `SIMULATION_RADIUS` and to base `SIMULATION_RADIUS_SECLIG` on the length of longest axis of the primary ligand to allow sampling of the secondary ligands around the primary ligand.

6.6 SILCS-MC: Ligand Optimization

6.6.1 Background

The power of SILCS lies in the ability to use FragMaps to rapidly evaluate binding of diverse ligands to a target. SILCS-MC is Monte-Carlo (MC) sampling of ligands in translational, rotational, and torsional space in the field of FragMaps. MC sampling uses CGenFF force field intramolecular energies and the Ligand Grid Free Energy (LGFE), which is the sum of atomic Grid Free Energies (GFEs). The Exclusion Map prevents ligand sampling where no probe or water molecules visited during SILCS simulations. SILCS-MC allows for rapid conformational sampling of the ligand while accounting for protein flexibility in a mean-field-like fashion since ligand affinity and volume exclusion are embedded in the combination of FragMaps and the Exclusion Map. Final ligand scoring is based on the LGFE score [15][16].

FragMaps previously generated for a target (see *SILCS Simulations*) can be used for optimization of a parent ligand by rapidly evaluating LGFE scores for derivatives of the parent ligand. The SILCS-MC approach has two principal advantages over free energy perturbation (FEP): functional group modifications to the parent ligand are very quick to evaluate, and there is no upper size limit on these modifications. Because the calculations are done in the context of pre-computed FragMaps, it is easy to decide which functional groups are candidates for modification simply by viewing the binding pose of the parent along with the FragMaps. FragMap densities adjacent to but unoccupied by parent ligand atoms offer opportunities for growing the parent ligand. Conversely, parent ligand atoms not in favorable regions of the FragMap densities can be candidates for modification or deletion without affecting binding affinity. Quantitative evaluation of individual functional group contributions to binding affinity can be achieved by analysis of atomic GFE scores. These capabilities are particularly useful when modifying functional groups to optimize pharmacokinetic properties or attempting to reduce the ligand size while maintaining affinity.

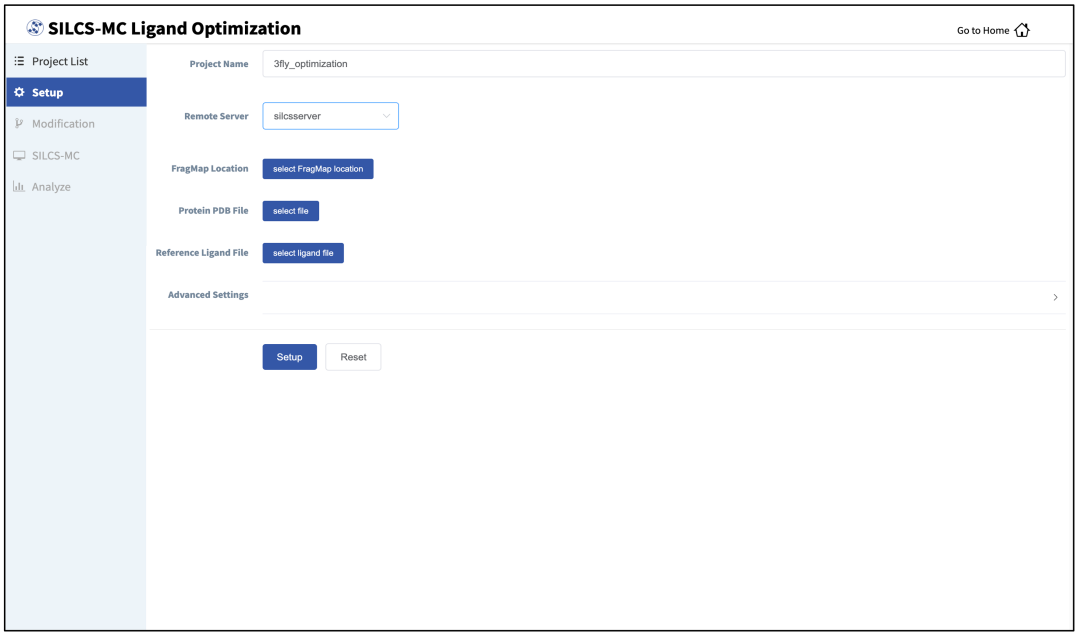
6.6.2 SILCS-MC Ligand Optimization Using the SilcsBio GUI

1. **Begin a new SILCS-MC Optimization project:**

   Select *New SILCS-MC Optimization project* from the Home page.

2. **Enter a project name, select the remote server, and input files:**

   Enter a project name and select the remote server where your SILCS-MC jobs will run. Next, provide FragMap, protein, and ligand input files. You may choose these files from the computer where you are running the SilcsBio GUI ("localhost") or from any server you have previously configured, as described in *File and Directory Selection*.



Tip: The "Reference Ligand File" must be an SDF or Mol2 format file that contains the parent ligand aligned in the binding pocket of the input protein.

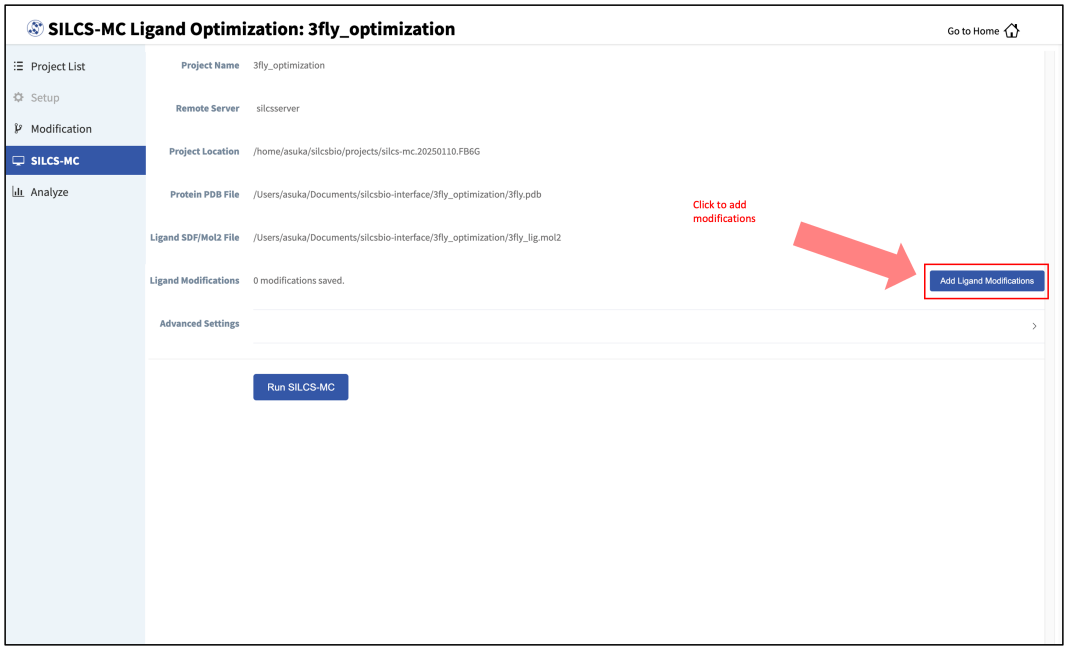
3. **Upload input files to the remote server:**

   Once all information is entered correctly, press the "Setup" button at the bottom of the page. The GUI will contact the remote server and upload the your input files to the "Project Location" directory on the remote server. A green "Setup Successful" button will appear once the upload has successfully completed. Press this button to proceed.

4. **Add ligand modifications:**

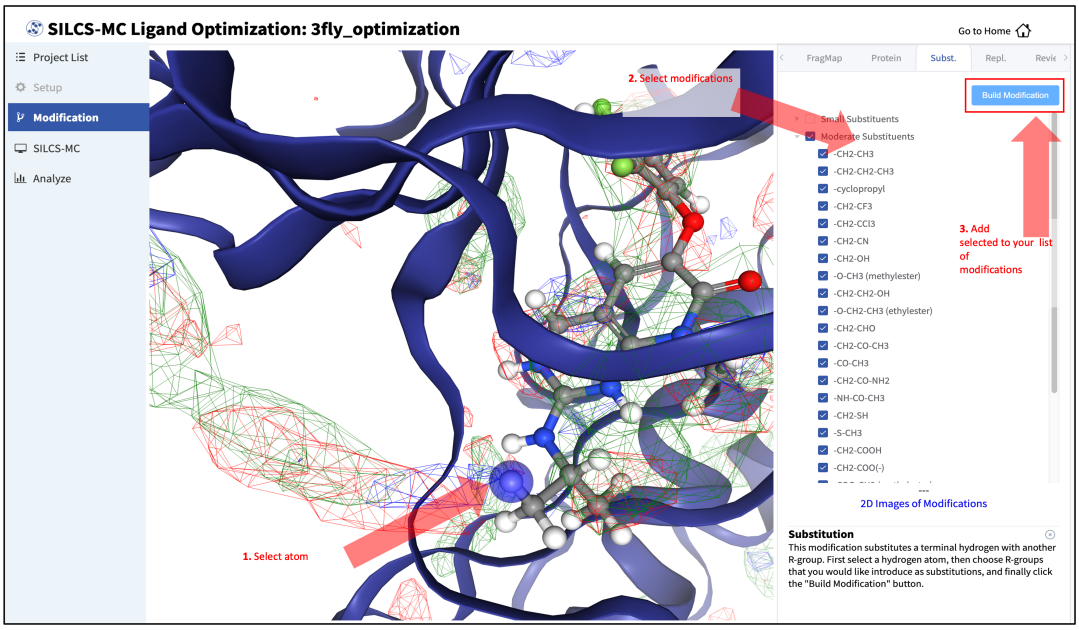
   The GUI will display a summary screen with the Project Name, Remote Server, Project Location, Protein PDB File, Ligand SDF/Mol2 File, and Ligand Modifications. The entry for Ligand Modifications will state "0 modifications saved" and have a button labeled "Add Lig-

and Modifications." Press this button to select modifications to the parent ligand for evaluation by SILCS-MC.



You will now see the parent ligand in the binding pocket. Rotate the view until you can easily see the functional group you wish to modify. There are two major modification types, Substitution and Replacement, available in the GUI. Substitution is used to substitute an atom with a functional group. Replacement is used to replace an atom in a ring or aliphatic acyclic group with another functional group that preserves the connectivity and valence of the ring or chain.

In the visualization window, select the atom to be modified. Then, select your desired modifications from the "Subst." (Substitution) or the "Repl." (Replacement) tab in the right-hand panel. Pressing the "Build Modification" button in the panel will take you to the "Review" tab.

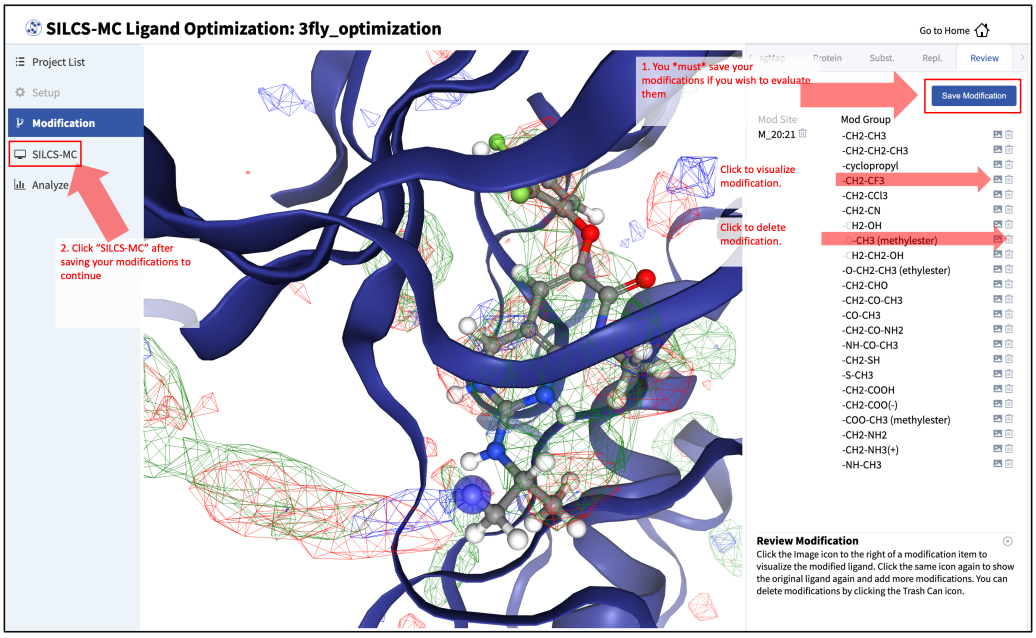


The list of modification types in the GUI covers a very broad range of chemical functionality and size.

5. **Confirm selected ligand modifications:**

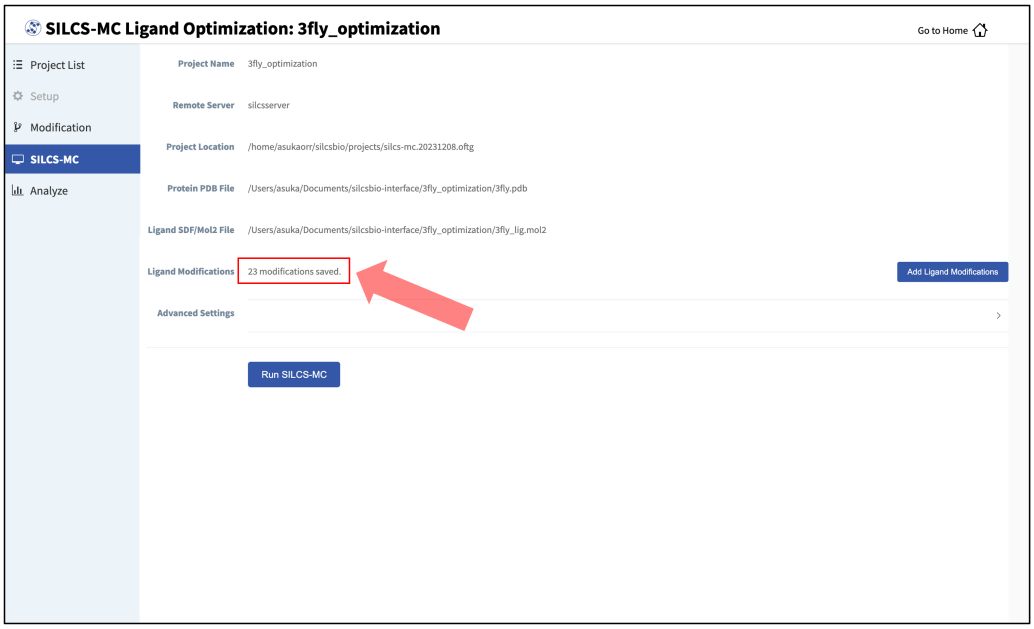
   Use the "Review" tab to confirm your desired modifications.

   Valid modifications will have a small Image icon as well as a small Trash Can icon. Clicking on the Image icon will show the modification in the center panel. Clicking on it again will show the parent ligand. Clicking on the Trash Can icon will delete the proposed modification from your list. You can go back to the "Subst." and "Repl." tabs to add to your list. Once you have completed your list of modifications, **you must press the "Save Modification" button in the "Review" tab to actually save the list of modifications for evaluation by SILCS-MC**.



6. **Launch SILCS-MC jobs:**

   Once you have saved all your desired modifications, click on *SILCS-MC* in the left-hand panel to go back to the summary screen. The "Ligand Modifications" will have been updated to reflect the number of saved modifications that will be evaluated by SILCS-MC. Because SILCS-MC is a fast calculation, it is feasible to select a very large number of modifications (hundreds+) for quick evaluation in a single run.



Tip: If you ran Halogen SILCS simulations for your target, you can include the Halogen

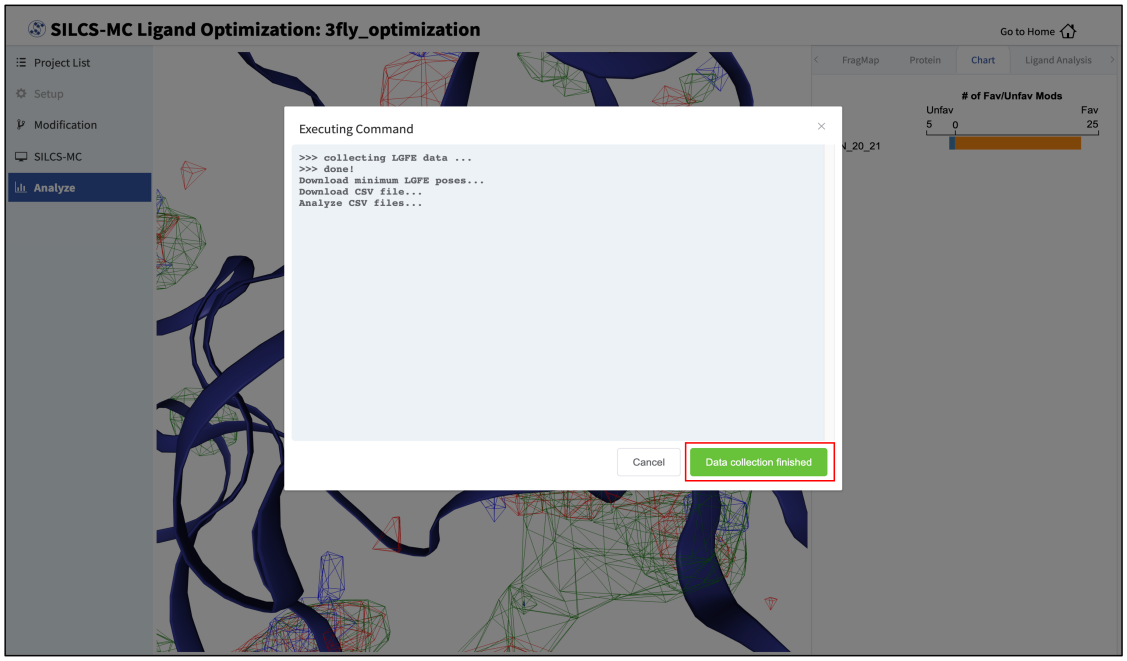
SILCS FragMaps in the SILCS-MC posing and scoring by checking the “Include Halogen FragMaps” box.

Click the “Run SILCS-MC” button to launch your jobs on the remote server.

You can monitor job progress in the SilcsBio GUI, and click on the “Show Chart” button once all job progress bars are at 100%.

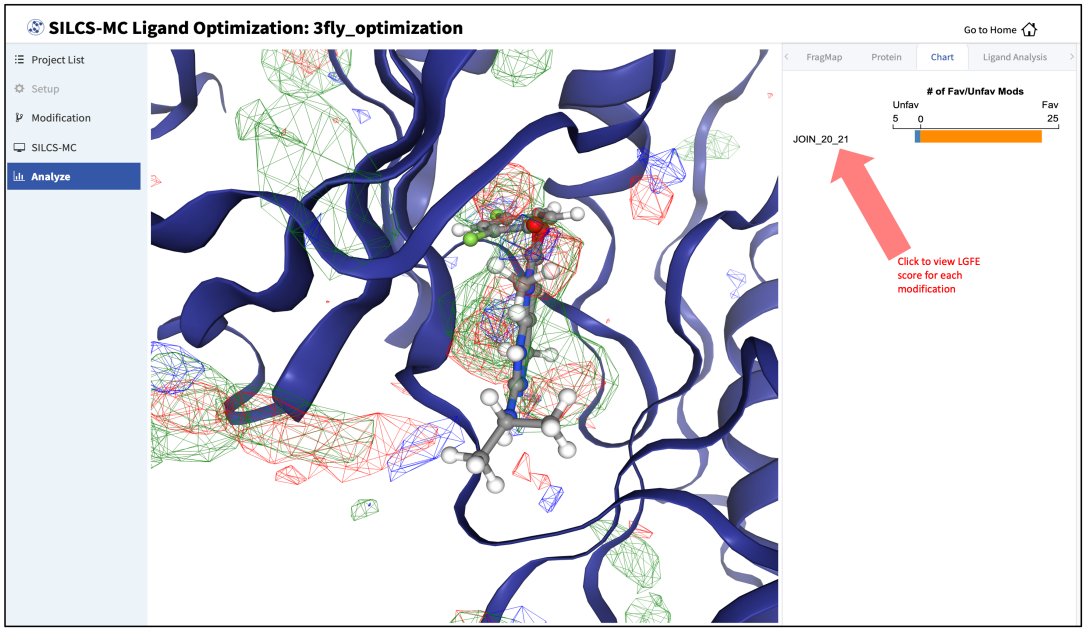
7. **Collect SILCS-MC results:**

   Clicking the “Show Chart” button after jobs have reached 100% will download the SILCS-MC results from the server. Click on “Data collection finished” to proceed.

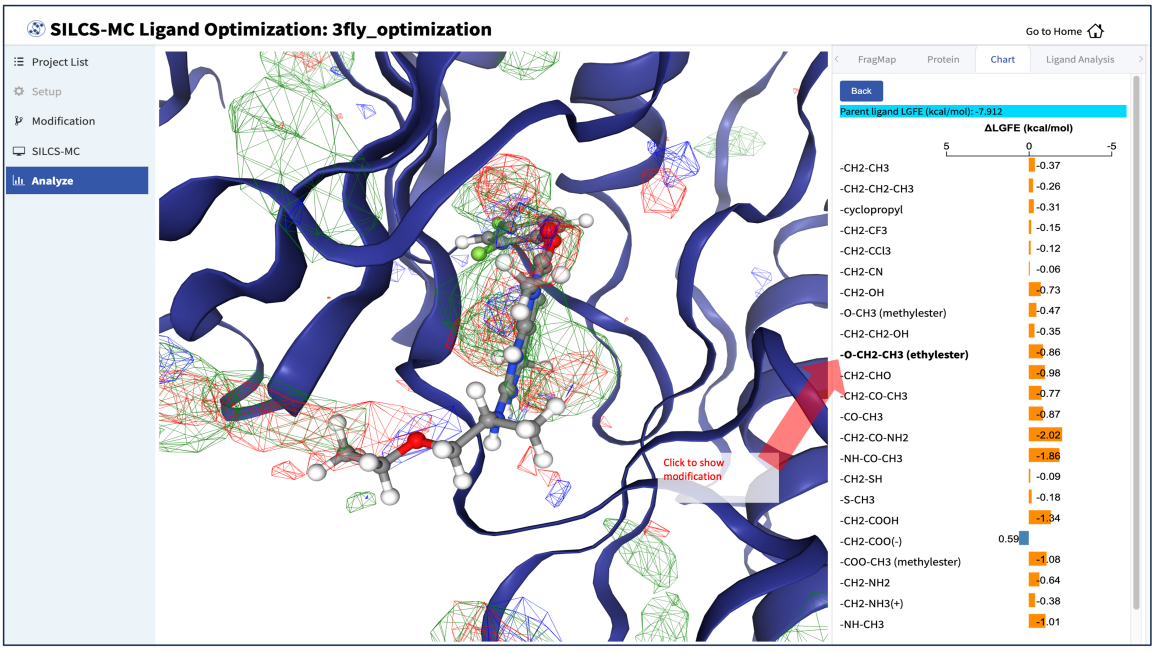


8. **Visualize ligand modifications and change in LGFE scores:**

The "Chart" tab in the right-hand panel shows a summary of the number of favorable and unfavorable modifications associated with a particular modification site. Click on the name of the modification site in that panel to see a detailed list of the change in Ligand Grid Free Energy ($\Delta LGFE$) relative to the parent ligand for each modification at that site.

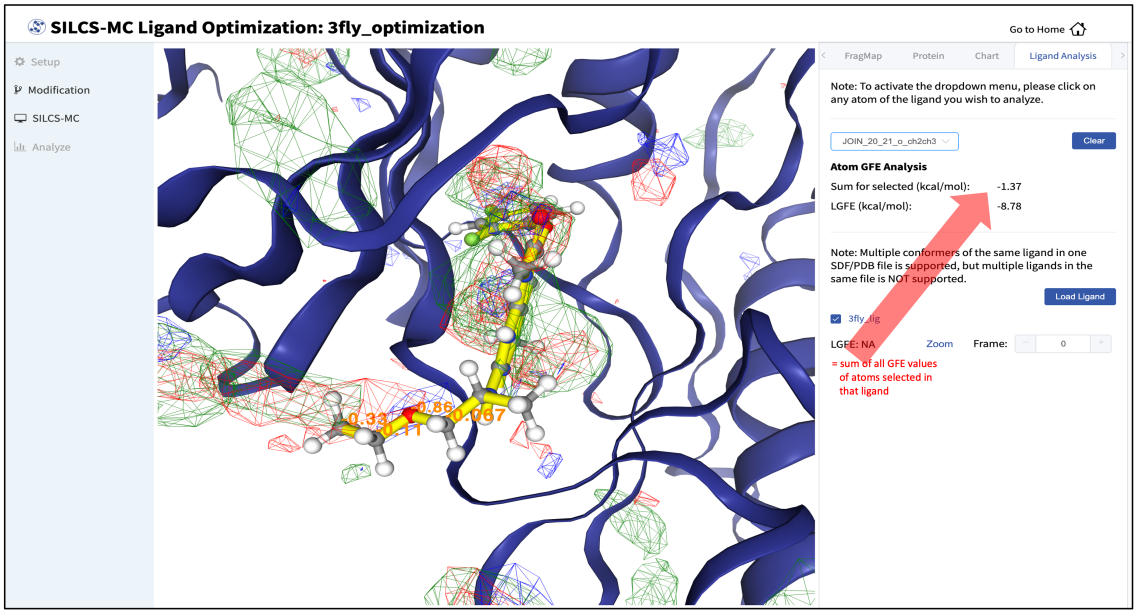


Clicking on the name of a modification will display that modification in the central panel.



The "Ligand Analysis" tab in the right-hand panel can be used to reveal the contribution of each scored atom in the ligand to the overall LGFE score. Clicking on a ligand atom in the central panel will display its atomic GFE score and add it to list of atoms that are used to

compute the "Sum" of GFE values for that ligand in the right-hand panel.



Detailed analysis of atomic GFE scores is an intuitive and powerful means to enable decisions regarding ligand modifications. Such analysis can reveal the affinity contribution of a new functional group. It can also reveal how the new functional group may have led to a change in the orientation of the ligand in the binding pocket relative to the parent compound. In such a case, comparison of atomic GFE scores for atoms preserved between the parent and derivative can reveal the preserved atoms' contributions to the change in binding pose. See Table 2 of [20] and Figure 6 of [21] for examples.

6.6.3 SILCS-MC Ligand Optimization Using the CLI

Running SILCS-MC ligand optimization from the command line interface consists of two steps: creating a set of Mol2 files with the desired modifications to the parent ligand and then running SILCS-MC pose refinement on these files. Details for these steps can be found in reference *Chemical Group Transformations* and in reference *SILCS-MC Pose Refinement Using the CLI*, respectively.

6.7 SILCS-BML

6.7.1 Background

Bayesian Machine Learning (BML) Markov Chain Monte Carlo optimization allows improved predictability of the LGFE from SILCS-MC docking or pose refinement through reweighting of the FragMaps when experimental data is available for compounds in a training set. As the standard FragMaps act as priors for the optimization, the number of compounds in the training set may be relatively small (i.e., as little as 10 compounds). The ability to use a small training set allows the approach to be used in early stage ligand optimization, enabling improved predictions, and facilitates the approach to be used and updated iteratively during the drug design process, which can involve splitting the compounds into congeneric series. SILCS-BML optimization involves reweighting of the FragMap contributions to the LGFE to better reproduce selected target data. BML reweighting can target the RMSD, Pearson correlation (R), or Percent Correct (PC) metrics. The reweighting should be performed in the presence of restraints, including flat-bottom (default), hard wall, or harmonic potentials. Output from the program is directly input into a new params file to be used in subsequent SILCS-MC docking. Reweighting includes k-fold cross validation. Note that the quality of the reweighted parameters is judged based on the results from the subsequent SILCS-MC docking. In cases of overfitting, the new LGFE scores will yield poor correlations. Additional details on SILCS-BML are available in references [16] and [18]

6.7.2 SILCS-BML Using the CLI

SILCS-BML is currently available only in CLI. BML is performed in five steps as follows:

1. **Generate fitting data:**

```
$SILCSBIODIR/silcs-bml/1_gen_fitting_data posepdbsdir=<location␣
↪and name of directory with docking poses as pdbs>
```

The above command will set up SILCS-BML runs by generating training and validation sets based on PDB files of docked ligands under investigation and a user-provided list of experimental binding affinity data for the same ligands. The docked ligands should correspond to those generated from SILCS-MC docking, located in the `minconfpdb` directory with GFE scores of the different classified atoms included in each PDB file.

Note: SILCS-BML requires pdbs generated from SILCS-MC docking and experimental data provided as a list of experimental binding affinity data (`exp.dat` file in the directory in which the `1_gen_fitting_data` command is being executed or filepath specified by the user as an *optional parameter*). The SILCS-MC docked poses in PDB format can be generated using the `silcs-mc/2_cals_lgfe_min_avg_sd` command with option `pdb=true`.

Required parameter:

- Path and name of directory containing ligand docked poses:

```
posepdbsdir=<location and name of directory with␣
↪docking poses as pdbs>
```

The ligand docked poses must be in PDB format derived from SILCS-MC docking. This directory will correspond to `3_silcsmc/minconfpdb` if SILCS-MC was implemented with default parameters as described in *SILCS-MC Docking Using the CLI*.

Optional parameters:

- Path and name of data file containing experimental binding affinities in kcal/mol:

```
exp=<experimental data in same alphanumeric order as␣
↪pose pdbs; default=exp.dat>
```

The binding affinity data must be listed in a single column text file with one value per line. The data should be ordered such that the binding affinities of the ligands are listed in the same order as the pdb file names of the same ligands in alphanumeric order. By default, the file name will be `exp.dat`.

- The number of sets used for cross-validation:

```
kfold=<k-fold validation; default=5>
```

The supplied data sample will be split into a number of sets for cross-validation. By default, `kfold=5`, and the sample data will be split into 5 sets.

Note: The 0th set `set=0` will always be prepared and it will use all data as training data for the SILCS-BML refinement.

- Path and name of output directory:

```
silcsbmldir=<location and name of output directory;␣
↪default=6_silcsbml>
```

- Option to skip the confirmation of the order of the experimental data:

```
confirm=<the order of the exp data has been confirmed;
↪ true/false; default=false>
```

When `confirm=false`, the ligand pdb file names will be printed on the terminal in the order in which the experimental data is expected to be (alphanumeric order). The user will be prompted to cancel or continue the command after viewing the

order of the ligands with either `y` to continue or `n` to cancel. If the user is confident in the order of the experimental data, this confirmation stage can be skipped with `confirm=true`.

2. **Run the refinement code:**

```
$SILCSBIODIR/silcs-bml/2_run_refine posepdbsdir=<location and␣
↪name of directory with docking poses as pdbs>
```

The `2_run_refine` command will execute the SILCS-BML fitting based on the setup from the previous step. The initial RMSD, Pearson correlation coefficient, and percent correct of the ligand LGFEs and experimental binding affinities will be printed on the terminal next to the final RMSD, Pearson correlation coefficient, and percent correct of the final LGFEs optimized by reweighting FragMap contributions targeting the user-defined metric (`target`). Additionally, the sets with the best RMSD (lowest RMSD), Pearson correlation coefficient (highest R), and percent correct (highest PC) will be printed on the terminal. This information will also be saved in `6_silcsbml/2_run_refine.out`.

Required parameters:

- Path and name of directory containing ligand docked poses:

```
posepdbsdir=<location and name of directory with␣
↪docking poses as pdbs>
```

The ligand docked poses must be in PDB format derived from SILCS-MC docking. This directory should correspond to the same directory specified in the first `$SILCSBIODIR/silcs-bml/1_gen_fitting_data` step.

Optional parameters:

- Path and name of output directory:

```
silcsbmldir=<location and name of output directory;␣
↪default=6_silcsbml>
```

The path and name of the output directory should correspond to the same path and name specified in the first `$SILCSBIODIR/silcs-bml/1_gen_fitting_data` step.

- Target metric for optimization (integer):

```
target=<choose optimization target (1: RMSD | 2:␣
↪Pearson Correlation | 3: Percent correct);␣
↪default=3>
```

SILCS-BML allows user to define the target property to be optimized with 3 options to choose from: Root mean square deviation (RMSD) with `target=1`, Pear-

son correlation (R) with `target=2`, or percent correct (PC) with `target=3`.

- Option to scale weights to yield original LGFE range (integer):

```
scale=<scale weights to yield original LGFE range␣
↪(1:Yes | 0:No); default=1>
```

The SILCS-BML fitting can be constrained such that the resulting LGFE values of the ligands are in the same range of the original set of LGFE values. This scaling can be turned off (`scale=0`) or kept on (`scale=1`).

- Restraint type:

```
restraint=<choose restraint type (1: Flat-bottom | 2:␣
↪Hard wall | 3: Harmonic); default=1>
```

Currently, 3 types of restraint are offered: flat-bottom (`restraint=1`), hard wall (`restraint=2`), and harmonic (`restraint=3`). By default, flat-bottom restraints are used (`restraint=1`).

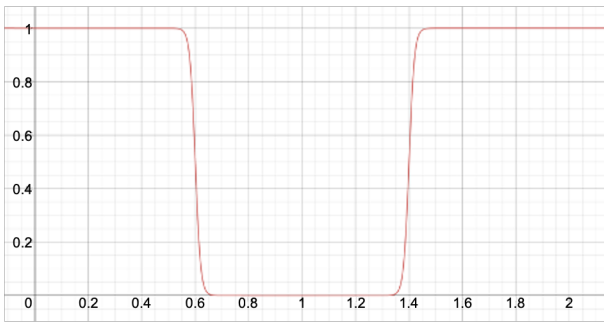
The restraint types are described in further detail below for weighting factor x, user defined force constant κ, and user-defined upper and lower limits of x, x_u and x_l, respectively. User-defined parameters κ, x_u, and x_l can be defined using the `kappa`, `upper`, and `lower` option bulletted below, after the description of restraint types.

[1]. Flat-bottom potential restraint (`restraint=1`)

$$y = \kappa\left(\frac{1}{1+\exp\frac{x-x_l}{d}}\right) \text{ if, } x < 1$$
$$y = \kappa\left(\frac{1}{1+\exp\frac{x_u-x}{d}}\right) \text{ if, } x > 1$$

Where d determines how flat the bottom is between x_l and x_u limits. The value of d is fixed at 0.01.

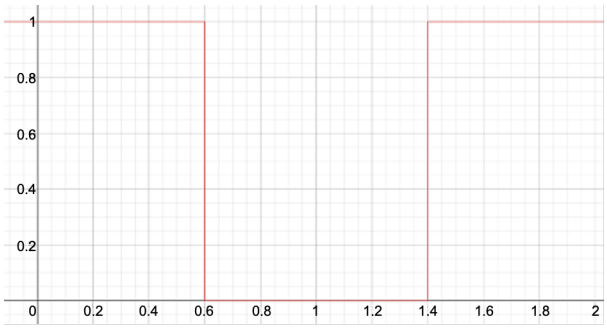
Below is a plot showing this restraint for $\kappa = 1$ (`kappa=1`), $x_l = 0.6$ (`lower=0.6`), and $x_u = 1.4$ (`upper=1.4`):



[2]. Hard-wall potential restraint (`restraint=2`)

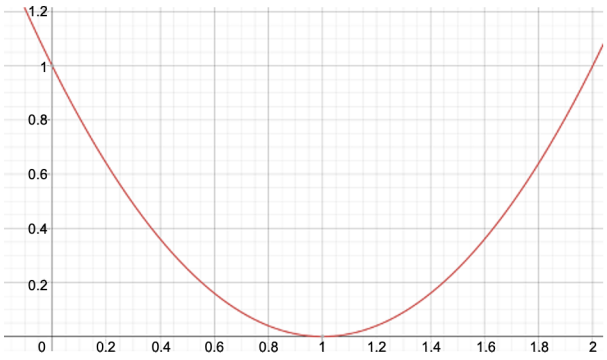
$$y = 0 \text{ if, lower} < \text{x} < \text{upper}$$
$$y = \kappa \text{ else}$$

Below is a plot showing this restraint for $\kappa = 1$ (`kappa=1`), $x_l = 0.6$ (`lower=0.6`), and $x_u = 1.4$ (`upper=1.4`):



[3]. Harmonic potential restraint (`restraint=3`)

$$y = \kappa(x - 1)^2$$



- Restraint force constant (integer):

```
kappa=<force constant for restraint; default=300>
```

- Upper limit of weighting factor (decimal):

```
upper=<upper limit of weighting factor for restraint␣
↪type 1 or 2; default=2>
```

- Lower limit of weighting factor (decimal):

```
lower=<lower limit of weighting factor for restraint␣
↪type 1 or 2; default=0.1>
```

- Minimum percentage of compounds with specific FragMap type to be included in optimization (decimal):

```
mincmpds=<minimum percentage of compounds with␣
↪specific FragMap type for that type to be included␣
↪in the optimization; default=30>
```

This option selects FragMap types to be fully optimized based on a minimum percentage of compounds with that specific FragMap type. FragMap types represented by less than the defined percentage will be optimized to a limited extent defined as (0.8~1.2) scaling of the input weight of that FragMap type. This prevents overfitting of underrepresented FragMaps. By default, `mincmpds=30`.

3. **Generate SILCS-MC Parameter File:**

```
$SILCSBIODIR/silcs-bml/3_gen_silcsmc_inp posepdbsdir=<location␣
↪and name of directory with docking poses as pdbs>
```

After determining the optimal FragMap weighting in the second step, the ligands can be re-docked using SILCS-MC parameter files reflecting the tuned FragMap weights. The SILCS-MC parameter file with the optimal FragMap weighting for each cross-validation set (`params_reweight_*.inp`) can be generated using the above command. The number of `params_reweight_*.inp` files will correspond to the number of cross-validation sets specified using the `k-fold` parameter in the first `1_gen_fitting_data` step.

Required parameter:

- Path and name of directory containing ligand docked poses:

```
posepdbsdir=<location and name of directory with␣
↪docking poses as pdbs>
```

The ligand docked poses must be in PDB format derived from SILCS-MC docking. This directory should correspond to the same directory specified in the `$SILCSBIODIR/silcs-bml/1_gen_fitting_data` and `$SILCSBIODIR/silcs-bml/2_run_refine` steps.

Optional parameters:

- Path and name of directory containing SILCS-BML fitting output:

```
silcsbmldir=<location and name of output directory;␣
↪default=6_silcsbml>
```

The path and name of the output directory should correspond to the same path and name specified in the previous two steps.

- Path and name of SILCS-MC parameter file:

```
paramsfile=<silcs-mc params.inp file to be reweighted;
↪ default=params.inp>
```

The `3_gen_silcsmc_inp` command will edit the `params.inp` file (typically `3_silcsmc/params.inp` from original SILCS-MC docking) that was used to dock and produce the poses stored in the directory specified for the `posepdbsdir` parameter. The new parameter files will be stored in `6_silcsbml` (or the user-specified `silcsbmldir`) as `params_reweight_*.inp`.

4. **Run SILCS-MC with the BML optimized FragMaps:**

```
$SILCSBIODIR/silcs-bml/4_run_silcsmc_bml prot=<prot PDB> ligdir=
↪<location and name of directory containing ligand mol2/sdf>
```

The above command will initiate SILCS-MC docking runs using the new parameter file(s) (`params_reweight_*.inp`) generated in the previous step, which specifies the reweighted FragMaps for the LGFE calculations. By default, the parameter set with the best percent correct will be used. The user can specify which parameter set to use with the optional `runset` parameter.

Required parameters:

- Path and name of target protein PDB file:

```
prot=<prot PDB>
```

- Path and name of directory containing ligand structure files:

```
ligdir=<location and name of directory containing␣
↪ligand mol2/sdf>
```

Typically, `ligdir` can be the same directory as the `posepdbsdir` parameter used in the previous `1_gen_fitting_data` step assuming the user has used the standard `3_silcsmc/minconfpdb` directory as the `posepdbsdir` parameter.

If the `ligdir` parameter is used, each molecule must be stored in its own individual file (Mol2 or SDF file) as only one molecule per file in the specified `ligdir` will be processed for SILCS-MC. If the user has an SD/SDF file containing multiple molecules, use the `sdfile` parameter instead of the `ligdir` parameter.

Optional parameters:

- Specify cross-validation set SILCS-MC parameter file (`params_reweight_*.inp`) to use:

```
runset=<set # to run silcs-mc with re-weighted␣
↪paramfile; All / BestPC / BestR / No. between [0-
↪kfold]; default=BestPC>
```

By default, the `paramsfile` originating from the cross-validation set with the best final percent correct will be used. The user can choose to specify which set to use with the optional `runset` parameter. The user can choose to perform SILCS-MC docking using the `params_reweight_*.inp` file(s) generated for:

- all (`runset=All`) cross-validation sets
- only the best percent correct (`runset=BestPC`) cross-validation set
- only the best Pearson correlation coefficient (`runset=BestR`) cross-validation set
- a specific cross-validation set (`runset=<any integer number from 0 to k-fold>`)

- Path and name of directory containing SILCS-BML fitting output:

```
silcsbmldir=<location and name of output directory;␣
↪default=6_silcsbml>
```

The path and name of the output directory should correspond to the same path and name specified in the previous three steps.

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing␣
↪FragMaps; default=maps>
```

- Path and name of SD file containing all ligands:

```
sdfile=<location and name of SD/SDF file>
```

- Option to bundle jobs:

```
bundle=<bundle multiple (single) jobs into larger␣
↪jobs; default=true>
```

- Number of jobs to bundle into one:

```
nproc=<number of jobs to bundle; default=8>
```

5. **Collect LGFE and poses from SILCS-MC simulations:**

```
$SILCSBIODIR/silcs-bml/5_calc_lgfe_bml ligdir=<location and name␣
↪of directory containing ligand mol2/sdf>
```

After the SILCS-MC docking simulations are complete, the above command will collect the resulting LGFE and docked poses of all ligands. The new RMSD, Pearson correlation coefficient, and percent correct of the re-docked ligands' LGFE scores versus the experimental binding affinities will be printed to the terminal. The following output files will be generated for the user to review:

- `6_silcsbml/report_bml_<runset>.csv` containing the same information as printed on the terminal
- `6_silcsbml/3_silcsmc_bml_<runset>/minconfpdb` containing the docked poses of the ligands
- `6_silcsbml/3_silcsmc_bml_<runset>/lgfe.csv` containing the new ligand LGFE scores
- `6_silcsbml/report_bml_<runset>.png` containing a correlation plot of the the experimental binding affinities versus ligand LGFE scores before and after SILCS-BML fitting

Required parameter:

- Path and name of directory containing ligand structure files:

  ```
  ligdir=<location and name of directory containing␣
  ↪ligand mol2/sdf>
  ```

  The same `ligdir` as the previous step should be used unless the user has specified `sdfile` instead of `ligdir` in the previous step.

Optional parameters:

- Specify cross-validation set SILCS-MC parameter file (`params_reweight_*.inp`) to use:

  ```
  runset=<set # to run silcs-mc with re-weighted␣
  ↪paramfile; All / BestPC / BestR / No. between [0-
  ↪kfold]; default=BestPC>
  ```

  The `runset` parameter should be the same as the previous `4_run_silcsmc_bml` step. Description of the `runset` parameter can be found in the previous step.

- Path and name of directory containint SILCS-BML fitting output:

  ```
  silcsbmldir=<location and name of output directory;␣
  ↪default=6_silcsbml>
  ```

The path and name of the output directory should correspond to the same path and name specified in the previous step.

- Path and name of SD file containing all ligands:

```
sdfile=<location and name of SD file>
```

- Number of top scoring poses to be collected:

```
npose=<number of best scoring poses sorted by LGFE;␣
↪default=1>
```

- Option to output poses in PDB format:

```
pdb=<write PDB file in minconfpdb folder; true/false;␣
↪default=false>
```

When the pdb keywoad is set to `true`, the top scoring poses will be saved in PDB format along with SDF file format. By default, `pdb=false`.

- Option to print ligand SMILES strings in output `6_silcsbml/3_silcsmc_bml_<runset>/lgfe.csv` file:

```
smiles=<write SMILES string to lgfe.csv;␣
↪default=false>
```

If `smiles=true` is used, the `6_silcsbml/3_silcsmc_bml_<runset>/lgfe.csv` output file will contain a column with the SMILCS string of each ligand.

- Calculate 4DBA descriptors and LGFE-4DBA scores:

```
fourdba=<calc 4 descriptors and LGFE-4DBA score; true/
↪false; default=false>
```

The bioavailablility of compounds can be evaluated using 4D Bioavailability (4DBA) scores. Compounds with higher 4DBA scores are more likely to have oral drug-like characteristics When `fourdba=true` additional calculations will be performed to calculate the 4DBA score for the ligands in addtion to LGFE score and LGFE-4DBA score. Please refer to *4D Bioavailability (4DBA) Calculation* for additional details on 4DBA scores.

- Path to python executable:

```
python=<path to python executable>
```

If the default path to python does not include the necessary modules, the path to the python executable with the necessary module installed must be specified with the `python` parameter. The default python path can be checked using the `which`

`python` command. Alternatively, entering the `$SILCSBIODIR/silcs-bml/5_calc_lgfe_bml` command will output the default path for python. The *matplotlib* package is required for plotting. If `fourdba=true`, then the *rdkit*, *tqdm*, *ipython* and *scipy* packages are also required. Please refer to *Python 3 Requirement* for more information.

Note: Note that once the new `params_reweight_*.inp` files with the optimized FragMap weighting factors for each cross-validation set is produced it may be used for SILCS-MC docking and pose refinement using the 1_run_silcsmc_custom command and specifying `paramsfile=params_reweight_<best performing cross-validation set>.inp` where `params_reweight_*.inp` is from step 3 (`3_gen_silcsmc_inp`) above.

6.8 SILCS-Water

6.8.1 Background

The SILCS-Water is a workflow that uses the SILCS-MC Docking protocol with water molecules as secondary ligands to identify the most favorable binding poses of ligands in presence of water molecules in the binding site of a protein. The water oxygen fragmaps (TIPO) are used to guide the docking of water molecules in the binding site.

SILCS-Water enables determination of the location of water molecules in the binding pocket and their impact on the predicted ligand binding orientation and affinities. The approach involves initial exhaustive MC docking of the ligand and 15 water molecules in a binding site followed by selection of a subset of waters for total LGFE calculation. In the approach, the Metropolis criteria for the MC is based on the overlap of the ligand and water with the SILCS FragMaps and the interaction energy between the waters and ligand. Results show that the SILCS-Water methodology is able to capture important waters and improve ligand positions and orientation. The workflow identifies waters interacting with ligands that occupy unfavorable locations with respect to the protein whose displacement through the appropriate ligands modifications should improve ligands binding affinity. More details of the validation study can be found in the paper [19].

6.8.2 SILCS-Water Using the SilcsBio CLI

Running SILCS-Water using the SilcsBio CLI basically involves running the SILCS-MC Docking/PosRef protocol with water molecules as secondary ligands by just adding the `seclig=true` option to the `1_run_silcsmc_*` commands. The default secondary ligands are 15 water molecules. The following steps outline the process of running SILCS-Water using the SilcsBio CLI where primary ligand is sampled for refinement and the waters are docked with exhaustive sampling. Users can also refer to the *SILCS-MC Docking Using the CLI* for more details.

1. **Launch SILCS-MC pose refinement runs with waters as secondary ligands:**

   To set up and run SILCS-MC pose refinement, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate `.mol2` or `.sdf` file. Alternatively, all the ligands can be combined into a single `.sdf` file. With this information, enter the following command to set up and launch SILCS-MC refinement runs with waters as secondary ligands:

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_local prot=<prot pdb> \
   ligdir=<directory containing ligand mol2/sdf> seclig=true
```

   Required parameters:

   - Path and name of protein PDB file:

```
prot=<protein pdb file>
```

   - Path and name of directory containing ligand SDF or Mol2 files:

```
ligdir=<location and name of directory containing ligand mol2/
↪sdf>
```

     If the ligdir option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. Subdirectories under `ligdir` will also be processed. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

     Warning: Ligands, regardless of file format, must include all hydrogens, including pH-appropriate (de)protonations, and must have reasonable three dimensional conformations.

   - Perform SILCS-MC docking with secondary ligands:

```
seclig=<true/false; default=false>
```

     By default, the SILCS-MC docking will be performed with the primary ligand only. Users can perform SILCS-MC docking simultaneously on multiple ligands by setting `seclig=true`. When set to true, 15 water molecules will be used as secondary ligands.

   For additional parameters and options:

   - Please refer to the *SILCS-MC Pose Refinement Using the CLI*.

2. **Evaluate docked poses:**

   The SILCS-MC tool set allows users to easily evaluate docked poses generated from the SILCS-MC docking runs. Users can rank order and collect the docked poses using `2_calc_lgfe_min_avg_sd` (see *Best-Pose Retrieval*).

For details on evaluating the resulting SILCS-MC docked poses, please refer to *Assessment of SILCS-MC Docked/Refined Poses in CLI*.

Since the water molecules were used as secondary ligands in the SILCS-MC docking, a selection criteria can be applied to determine which waters to include in the total LGFE calculation. The best poses will be selected based on the total LGFE after all the SILCS-MC docking runs have been analyzed and reported in the output `lgfe.csv` file. The default selection criteria is for water molecules as secondary ligands and it is adopted from the SILCS-WATER paper [19].

- Select secondary ligands based on distance cutoff:

```
distcutoff=<cutoff for max distance to select seclig in␣
↪Angstroms; default=4.0>
```

  The `distcutoff` parameter is the maximum distance allowed between the ligand and secondary ligand. By default, the `distcutoff` is set to 4.0 Å.

- Select secondary ligands based on LGFE cutoff:

```
lgfecutoff=<cutoff for max allowed LGFE of seclig (SLigLGFE);␣
↪default=0.5>
```

  The `lgfecutoff` parameter is the maximum LGFE allowed for the secondary ligand. By default, the `lgfecutoff` is set to +0.5 kcal/mol.

- Select secondary ligands based on total energy cutoff:

```
totecutoff=<cutoff for max allowed TotE (IntE+SLigLGFE) of␣
↪seclig; default=-0.6>
```

  The `totecutoff` parameter is the maximum total energy allowed for the secondary ligand. By default, the `totecutoff` is set to -0.6 kcal/mol.

- Maximum number of runs expected in the pdb files:

```
nrunsilcsmc=<max number of runs expected in the pdb files;␣
↪default=1250>
```

- Select secondary ligands based on interaction energy cutoff:

```
intecutoff=<cutoff for max allowed interaction energy (IntE)␣
↪between lig-seclig; default=-1.1>
```

  The `intecutoff` parameter is the maximum interaction energy allowed between the ligand and secondary ligand. By default, the `intecutoff` is set to -1.1 kcal/mol. Note that the interaction energy is the sum of the electrostatic and Lennard-Jones interaction energies between the ligand and the secondary ligand to be selected.

- Sort the output by total LGFE:

```
sortbytotlgfe=<sort lgfe.csv output by TotLGFE; true/false;␣
↪default=false>
```

When `sortbytotlgfe=true`, the output `lgfe.csv` file will be sorted by the total LGFE in ascending order. By default, the output `lgfe.csv` file is sorted by the ligand name in alphanumeric order.

CHAPTER

SEVEN

COVALENT SUITE

7.1 FragMap Visualization

For more information on visualizing FragMaps, please refer to *FragMap Visualization* in the *Small Molecule Suite*.

7.2 SILCS Simulations

For more information on running SILCS simulations, please refer to *SILCS Simulations* in the *Small Molecule Suite*.

7.3 SILCS-MC: Docking and Pose Refinement

For more information on SILCS-MC docking and pose refinement, please refer to *SILCS-MC: Docking and Pose Refinement* in the *Small Molecule Suite*.

7.4 SILCS-MC: Ligand Optimization

For more information on SILCS-MC ligand optimization, please refer to *SILCS-MC: Ligand Optimization* in the *Small Molecule Suite*.

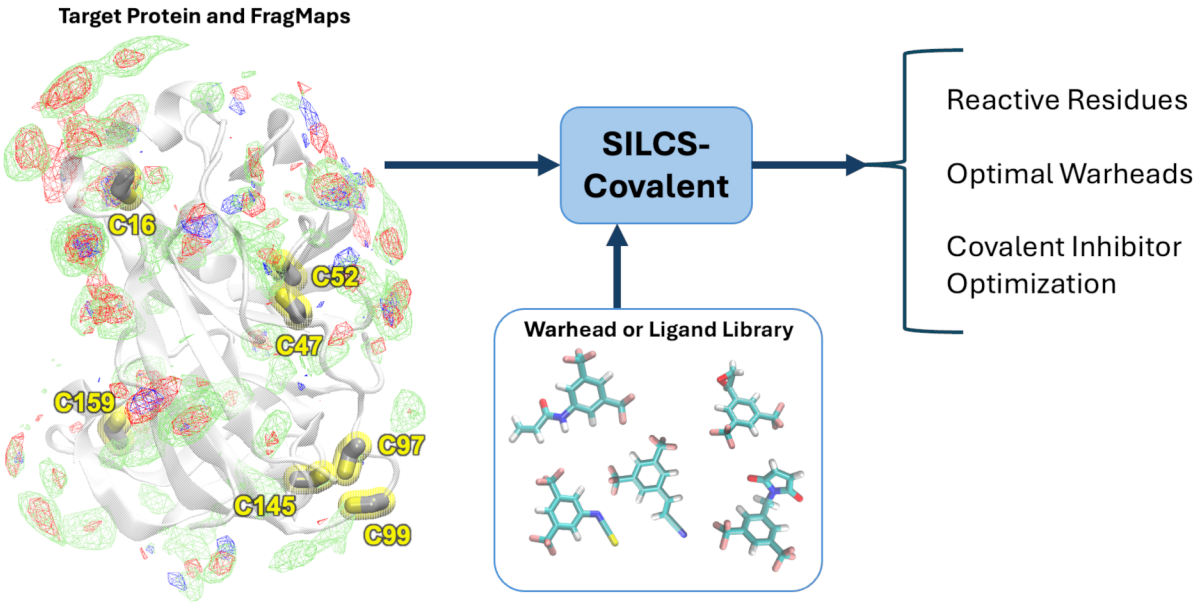
7.5 SILCS-BML

For more information on SILCS-BML and its usage, please refer to *SILCS-BML* in the *Small Molecule Suite*.

7.6 SILCS-Covalent

7.6.1 Background

The SILCS-Covalent workflow is a comprehensive approach to covalent drug design that combines molecular simulations, fragment-based binding site identification, and machine learning models. This workflow addresses the challenges of designing covalent drugs by providing insights into the binding preferences and reactivity of potential covalent ligands. It utilizes SILCS FragMaps to identify reactive cysteine residues and select optimal warhead groups. Other covalent linkages are also supported. Machine learning models trained on activity data further aid in predicting the efficacy and selectivity of warheads. The workflow provides the efficiency and reliability in identifying potential covalent ligands and optimizing their non-covalent interactions. Additional details on SILCS-Covalent are available in reference [22].



Note: The SILCS-Covalent workflow is slightly different from the *SILCS-MC Docking in Restrained Mode* workflow. The *SILCS-MC Docking in Restrained Mode* workflow is designed to run the SILCS-MC Docking protocol with harmonic restraint on selected atoms of the ligand. The SILCS-Covalent workflow is designed to run the SILCS-MC docking without any restraints on the reactive atom of the ligand first and then run the analysis to identify the ligand conformations with reactive atoms located within a certain distance from the reactive atoms of the residues in

the binding site. The SILCS-Covalent workflow allows the reactivity of the target residues and the warheads or ligands to be assessed, whereas the *SILCS-MC Docking in Restrained Mode* workflow only identifies energetically favored binding poses given that a covalent bond is formed between a target residue and the ligand.

7.6.2 SILCS-Covalent Using the SilcsBio CLI

1. **Launch SILCS-Covalent docking runs:**

   To set up and run SILCS-Covalent docking from the command line interface, create a directory containing all the ligands to be evaluated. Each ligand can be stored as a separate SDF or Mol2 file. Alternatively, all the ligands can be combined into a single SDF file. With this information, enter the following command to set up and launch SILCS-Covalent docking runs:

   ```
   ${SILCSBIODIR}/silcs-covalent/1_run_silcs_covalent prot=<prot␣
   ↪pdb>
   ```

   Required parameters:

   - Path and name of protein PDB file:

     ```
     prot=<protein pdb file>
     ```

   Optional parameters:

   - Mode of the SILCS-Covalent workflow:

     ```
     mode=< warhead or ligand >
     ```

     The mode of the SILCS-Covalent worflow can be set to either `warhead` or `ligand`. The default mode is `warhead`.

     – In `ligand` mode, the workflow will only run the SILCS-MC Docking protocol on the ligands provided to identify the most favorable binding poses of the ligands in the binding site around the linking residue(s). In the `ligand` mode, the user must provide the ligands in SD file or Mol2 format using either the `ligdir` or `sdfile` option.

     – In `warhead` mode, the workflow will run the SILCS-MC Docking protocol on the warheads provided to identify the most favorable binding poses of the warheads in the binding site around the linking residue.

       Additionally in `warhead` mode, the user may provide additional information on the experimental data of the warheads such as kGSH, the glutathione (GSH) reactivity rate constant. The experimental data can be used to train

machine learning models to quantify the effectiveness of various warhead groups for covalent drug design for the given linking residue of your target protein. Currently, this feature is only available for the cysteine (CYS) residues as the linking residue and an ML model trained on the Petri's Warhead Library [23] will be used to predict the activity class for the warheads specific to the linking cysteine residue.

Please refer to reference [22] for more details.

- Path and name of directory containing ligand SD or Mol2 files:

```
ligdir=<ligand directory>
```

If the `ligdir` option is used, only one molecule per file under `ligdir` will be processed for SILCS-MC docking. For an SD/SDF file containing multiple molecules, use `sdfile=<path to sdfile>` instead of the `ligdir` option.

- Path and name of output directory for the SILCS-Covalent workflow:

```
covaldir=<output directory default=4_covalent_warhead>
```

- The residue name of the amino acid for covalent link with the ligand:

```
linkresname=<residue name for covalent link; CYS/LYS/SER/THR;
↪ default=CYS>
```

Currently, only cysteine (CYS), lysine (LYS), serine (SER), and threoine (THR) residues are supported. Asparagine (ASN) residues will be supported in the future.

- The residue ID of the bound amino acid for covalent bonding with the ligand:

```
linkresid=<residue ID for covalent link; default=NULL>
```

By default, SILCS-MC docking of each warhead or ligand will be performed in proximity to each residue with residue names specified by `linkresname` or each cysteine of the target protein if the `linkresname` option is not used. If the `linkresid` option is specified, then the SILCS-MC docking will only be performed in proximity to the residue with the residue ID specified by `linkresid` will be targeted.

- The residue ID(s) of amino acids disqualified from covalent bonding with the ligand:

```
skipresid=<residue IDs to skip; e.g. "145,231"; default=NULL>
```

Certain residues may be ommitted from consideration in the SILCS-MC docking by specifying their residue IDs with the `skipresid` option. This option can be

useful in omitting docking calculations near cysteine residues involved in disulfide bonds.

- Path and name of directory containing FragMaps:

```
mapsdir=<location and name of directory containing FragMaps;␣
↪default=maps>
```

The supplied `mapsdir` should contain the protein side chain probability maps. The probability maps for each type of residue are generated by default during the `silcs/2b_gen_maps` and `silcs/2c_fragmap` steps in the latest version of the SilcsBio software. Probability maps for a specific residue ID can be generated by re-running the `silcs/2b_gen_maps` and `silcs/2c_fragmap` steps with the `residmap` option. Please refer to *SILCS Simulations Using the CLI* for more details.

- Use GPU for the SILCS-MC Docking protocol:

```
gpu=<true/false; default=true>
```

When `gpu=false` and if the `bundle` option is set to true, the workflow will run the SILCS-MC Docking protocol in parallel using multiple processors. The number of processors can be specified using the `nproc` option. The default value for `nproc` is the number of processors available on the system.

When `gpu=true`, the `bundle=true` option works differently. The `bundle` option will bundle `nproc` number of SILCS-MC Docking runs into a single job. The default value for `nproc` is the number of processors available on the system.

```
bundle=<true/false; default=false>
nproc=<# of processor used when bundle=true>
```

- Translation distance for defining sampling spheres in six directions around the reactive atom of the linking residue in Å:

```
translation=<translation distance for defining sampling␣
↪spheres; default=3.5>
```

- Path and name of the custom parameters file for the SILCS-MC Docking protocol:

```
paramsfile=<custom params file>
```

By default, the SILCS-MC Docking protocol in the SILCS-Covalent workflow uses the parameters file located in `${SILCSBIODIR}/templates/silcs-covalent/params-<warhead/ligand>.tmpl`.

2. **Evaluate docked poses:**

After the SILCS-Covalent docking runs are completed, the docked poses are evaluated to identify the most favorable binding poses of the ligands in the binding site around the linking residue.

To evaluate the docked poses, enter the following command:

```
${SILCSBIODIR}/silcs-covalent/2_calc_lgfe_probres prot=<prot pdb>
↪ covaldir=<name of output directory>
```

The above command will evaluate the poses based on three criteria:

- `eq` : Best pose when sorted by LGFE score
- `rx` : Best pose when sorted by ProbRes score with LGFE < 0 kcal/mol
- `rxeq` : Best pose when sorted by LGFE score with a ProbRes score greater than the ProbRes cutoff value. This pose will have both a favorable LGFE score and potential for covalent bond formation.

Required parameters:

- Path and name of protein PDB file:

```
prot=<protein pdb file>
```

- Path and name of output directory for the SILCS-Covalent workflow:

```
covaldir=<output directory default=4_covalent_warhead>
```

Optional parameters:

- Path and name of the file containing the list of warhead/ligand name and reactive atom IDs:

```
whidfile=<file with id list for warhead/ligand reactive atom;
↪ default="Petri's Warhead Lib">
```

The file specified by the `whidfile` option should contain two columns. In the first column, the warhead or ligand names are listed. In the second column, the reactive atom IDs of the warheads or ligands are listed:

> Each line should have two columns: `<warhead_name/ligand_name>` `<reactive_atom ID>`

By default, the `whidfile` for Petri's Warhead Library is used. The Petri's Warhead Library `whidfile` can be found in `${SILCSBIODIR}/data/databases/petri_warhead.whid`.

- Distance cutoff for covalent bond definition in Å:

```
dcutoff=<distance cutoff for covalent bond definition;␣
↪default=2.0>
```

The reactive atoms of the ligand and linking residue for a given docked pose are considered to be sufficiently close to form a covalent bond if the distance between the two atoms is within the cutoff distance defined by the `dcutoff` option.

- ProbRes cutoff for filtereing poses unlikely to form covalent bonds:

```
prcutoff=<cutoff value for ProbRes; default=0.025>
```

The probability of the reactive atoms of the ligand and linking residue for a given docked pose to be within the cutoff distance defined by `dcutoff` is represented by ProbRes. Poses that do not have ProbRes larger than the cutoff defined by the `prcutoff` are filtered from further analysis.

- Number of output poses:

```
npose=<number of best scoring poses; default=1>
```

By default, only three the best scoring, based on LGFE (`eq`), ProbRes (`rx`), and PRCutoff (`rxeq`), docked poses will be output. To output additional docked poses, use the `npose` option and specify the desired number of docked poses. E.g., if 5 poses are desired, using `npose=5` will result in the 5 best scoring poses being output for each warhead or ligand and each basis.

- Option to output docked poses in PDB format:

```
pdb=<true/false; default=false>
```

When `pdb=true` the docked poses will be output in PDB format in addition to the default SD file format.

- Option to calculate 4DBA descriptors, and 4DBA and LGFE-4DBA scores:

```
fourdba=<true/false; default=false>
```

When `fourdba=true` the four 4DBA descriptors of each input ligand will be calculated and output into `lgfe-4dba.csv`. A high 4DBA score indicates that the ligand is bioavailable while a low 4DBA score indicates that the ligand is not bioavailable. For more information of 4DBA and LGFE-4DBA scores, please see *4D Bioavailability (4DBA) Calculation*. The `rdkit`, `tqdm`, `ipython` and `scipy` packages must be installed for the 4DBA score calculation. Please refer to *Python 3 Requirement* for more information.

- Submit job to queuing system:

```
batch=<submit job to queuing system; true/false;␣
↪default=false>
```

- Path to the Python executable:

```
python=<path to python executable; default=${python}>
```

The default python path can be checked using the `which python` command.

7.6.3 Assessment of SILCS-Covalent Results in CLI

The `silcs-covalent/2_calc_lgfe_probres` command will generate a number of output files that can be used to identify the residues most ammenable to covalent bonding, identify the optimal warheads or ligands for a target protein or specific residues within a target protein, or optimize covalent inhibitors.

In both `warhead` and `ligand` modes, three poses in SD file format are output into `<covaldir>/<reactive residue>/minconfpdb/` for each warhead or ligand, where `<reactive residue>` is the linking residue in proximity to which the warhead or ligand was docked. The three poses correspond to the most favorable poses based on LGFE (`eq`), ProbRes (`rx`), and PRCutoff (`rxeq`):

- `<ligand|warhead>.eq.sdf` : Pose with the most favorable energy (lowest LGFE).
- `<ligand|warhead>.rx.sdf` : Pose with the highest probability of ligand and linking residue reactive atoms being in proximity (highest ProbRes) with favorable energy (LGFE < 0 kcal/mol).
- `<ligand|warhead>.rxeq.sdf` : Pose with the most favorable energy (lowest LGFE) with the probability of the ligand and linking residue reactive atoms being in proximity greater than a cutoff value (ProbRes > 0.025 by default). This pose will have both a favorable LGFE score and potential for forming a covalent bond with the reactive residue.

In addtion, for each reactive residue in proximity to which the warheads or ligands were docked, a summary of the LGFE and ProbRes of each of the three warhead or ligand poses (`eq`, `rx`, `rxeq`) will be output in `<covaldir>/<reactive residue>.<ligand|warhead>.csv`. Below are the contents of each column of `<covaldir>/<reactive residue>.<ligand|warhead>.csv`:

- In the first column, the warhead or ligand names are listed.
- In the second, third, and fourth columns, the LGFEs of the most favorable poses of the corresponding warhead or ligand in terms of LGFE (`eq`), ProbRes (`rx`), and PRCutoff (`rxeq`), respectively, are listed.
- In the fifth, sixth, and seventh columns, the ProbRes values of the most favorable poses of the corresponding warhead or ligand in terms of LGFE (`eq`), ProbRes (`rx`), and PRCutoff (`rxeq`), respectively, are listed.

In `warhead` mode with cysteine residues specified as the linking residue and the Petri's Warhead Library [23] specified as the warheads, in addition to the output results described above, predicted kGSH, the GSH reactivity rate constant, and activity classifications are output in `<reactive cysteine>.rf-ml.predicted.csv`, where `<reactive cysteine>` is the linking cysteine in proximity to which the warheads were docked. This machine learning based prediction of reactivity is not currently available for other residues beyond the default cysteine or other warhead libraries. If you wish to apply the machine learning algorithm to other residues and/or warhead libraries, please contact us at support@silcsbio.com. Below are the contents of each column of `<reactive cysteine>.rf-ml.predicted.csv`:

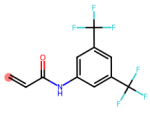
- In the first column, the warhead names are listed.
- In the second and third columns, the LGFEs of the most favorable poses of the corresponding warhead in terms of LGFE (`eq`) and ProbRes (`rx`), respectively, are listed.
- In the fourth column the ProbRes value of the most favorable pose of the corresponding warhead in terms of ProbRes (`rx`) is listed.
- In the fifth column, the predicted kGSH of the corresponding warhead for the linking cysteine is listed.
- In the sixth column, the predicted activity class of the corresponding warhead is listed, where `L` indicates low activity and `H` indicates high activity.

7.6.4 Default Warhead Library

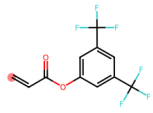
The Petri's Warhead Library [23] is the default library of warheads.

The Petri's Warhead Library structures can be found in `${SILCSBIODIR}/data/databases/petri_warhead`. The reactive atoms of each warhead can be found in the corresponding `whidfile` in `${SILCSBIODIR}/data/databases/petri_warhead.whid`.

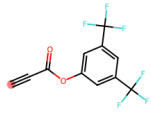
Below are the 2D structures of the Petri's Warhead Library with the reactive atoms highlighted in red:



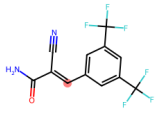
N-(3,5-Bis(trifluoromethyl)phenyl) acrylamide (1)



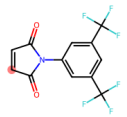
3,5-Bis(trifluoromethyl)phenyl acrylate (2)



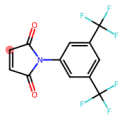
3,5-Bis(trifluoromethyl)phenyl propiolate (3)



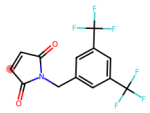
3-(3,5-Bis(trifluoromethyl)phenyl)-2-cyanoacrylamide (4)



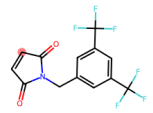
1-(3,5-Bis(trifluoromethyl)phenyl)-1H-pyrrole-2,5-dione (5)



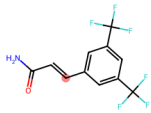
1-(3,5-Bis(trifluoromethyl)phenyl)-1H-pyrrole-2,5-dione (5_1)



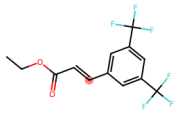
1-(3,5-Bis(trifluoromethyl)benzyl)-1H-pyrrole-2,5-dione (6)



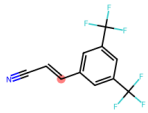
1-(3,5-Bis(trifluoromethyl)benzyl)-1H-pyrrole-2,5-dione (6_1)



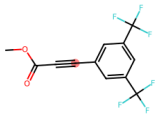
3-(3,5-Bis(trifluoromethyl)phenyl) acrylamide (7)



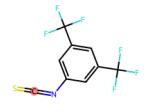
Ethyl-3-(3,5-bis(trifluoromethyl)phenyl) acrylate (8)



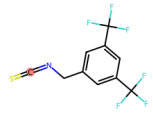
3-(3,5-Bis(trifluoromethyl)phenyl) acrylonitrile (9)



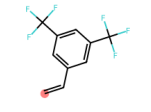
Methyl-3-(3,5-bis(trifluoromethyl) phenyl)propiolate (10)



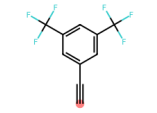
3,5-Bis(trifluoromethyl)phenyl isothiocyanate (11)



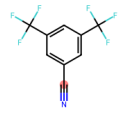
1-(Isothiocyanatomethyl)-3,5-bis(trifluoromethyl)benzene (12)



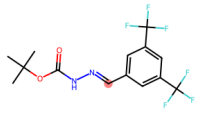
3,5-Bis(trifluoromethyl) styrene (13)



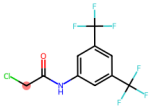
3,5-Bis(trifluoromethyl)phenyl acetylene (14)



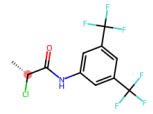
3,5-Bis(trifluoromethyl) benzonitrile (15)



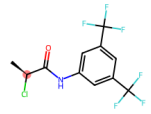
Tert-butyl-2-(3,5-bis(trifluoromethyl) benzylidene)hydrazinecarboxylate (16)



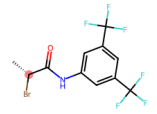
N-(3,5-Bis-trifluoromethyl-phenyl)-2-chloro-acetamide (17)



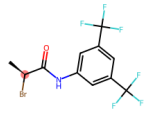
(R)-N-(3,5-Bis(trifluoromethyl) phenyl)-2-chloro-propanamide (18r)



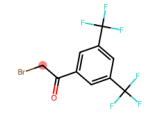
(S)-N-(3,5-Bis(trifluoromethyl) phenyl)-2-chloro-propanamide (18s)



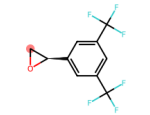
(R)-N-(3,5-Bis(trifluoromethyl) phenyl)-2-bromo-propanamide (19r)



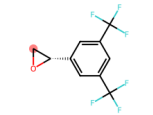
(S)-N-(3,5-Bis(trifluoromethyl) phenyl)-2-bromo-propanamide (19s)



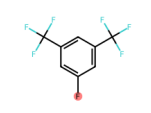
1-(3,5-Bis(trifluoromethyl) phenyl)-2-bromoethanone (20)



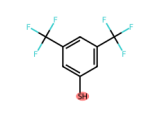
(R)-2-(3,5-Bis(trifluoromethyl) phenyl)oxirane (21r)



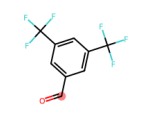
(S)-2-(3,5-Bis(trifluoromethyl) phenyl)oxirane (21s)



3,5-Bis(trifluoromethyl) fluorobenzene (22)



3,5-Bis(trifluoromethyl) benzenethiol (23)



3,5-Bis(trifluoromethyl) benzaldehyde (24)

CHAPTER

EIGHT

KINETICS SUITE

8.1 FragMap Visualization

For more information on visualizing FragMaps, please refer to *FragMap Visualization* in the *Small Molecule Suite*.

8.2 SILCS Simulations

For more information on running SILCS simulations, please refer to *SILCS Simulations* in the *Small Molecule Suite*.

8.3 SILCS-MC: Docking and Pose Refinement

For more information on SILCS-MC docking and pose refinement, please refer to *SILCS-MC: Docking and Pose Refinement* in the *Small Molecule Suite*.

8.4 SILCS-MC: Ligand Optimization

For more information on SILCS-MC ligand optimization, please refer to *SILCS-MC: Ligand Optimization* in the *Small Molecule Suite*.

8.5 SILCS-BML

For more information on SILCS-BML and its usage, please refer to *SILCS-BML* in the *Small Molecule Suite*.

8.6 SILCS-Hotspots

For more information on SILCS-Hotspots, please refer to *SILCS-Hotspots* in the *Small Molecule Suite*.

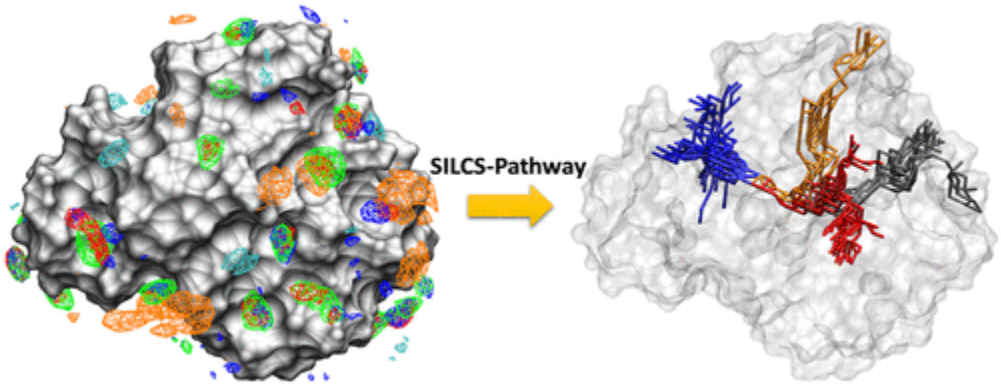
8.7 SILCS-Pathway

8.7.1 Background

SILCS-Pathway is a computational workflow designed to identify and analyze ligand dissociation pathways from protein or RNA binding sites. Unlike traditional approaches that focus solely on binding affinity, SILCS-Pathway leverages the correlation between drug efficacy and dissociation kinetics. The method builds on the Site Identification by Ligand Competitive Saturation (SILCS) technique, which precomputes FragMaps—free energy landscapes for various chemical functionalities around a target biomolecule.

SILCS-Pathway utilizes the A* pathfinding algorithm to enumerate all possible ligand dissociation routes from the binding site to the bulk solvent. The search is performed on a set of evenly distributed points around the protein, generated using a Fibonacci lattice. The cost function for the A* algorithm incorporates SILCS exclusion maps and grid free energy scores, enabling the identification of pathways that reflect local protein flexibility and favorable ligand interactions.

By systematically traversing all bulk solvent points, SILCS-Pathway locates and clusters all feasible dissociation pathways, providing a comprehensive view of ligand unbinding mechanisms. The approach has been validated against proteins previously studied with enhanced sampling molecular dynamics, demonstrating its ability to efficiently capture key dissociation routes. The identified pathways serve as a foundation for subsequent determination of ligand dissociation kinetics using SILCS free energy profiles. Additional details on SILCS-Pathway are available in reference [24].



8.7.2 SILCS-Pathway Using the SilcsBio CLI

There are six main steps in the SILCS-Pathway workflow, each of which can be executed using the SilcsBio CLI. Below is a detailed description of each step, including usage instructions, required and optional parameters, expected outputs, and important notes for successful execution.

To run all six steps of the SILCS-Pathway workflow sequentially, use the following command:

```
${SILCSBIODIR}/silcs-pathway/run_silcs_pathway prot=<protein PDB file>␣
↪center="x,y,z"
```

This script automates the complete SILCS-Pathway process, executing each step in order and handling intermediate files as needed. It is recommended for users who want a streamlined, end-to-end workflow.

Required parameters

- `prot`: Full path to the protein PDB file.
- `center`: Coordinates (x,y,z) of the binding site center (e.g., center="20.1,33,-2")(default: center.csv).

Optional parameters

- `mapsdir`: Directory containing FragMaps (default: maps).
- `outputdir`: Output directory for results (default: 4_pathway).
- `batch`: Submit jobs to queuing system (default: true).

Alternatively, you may execute each step individually as described below:

1. **Fibonacci Sphere Generation**

   This step generates a defined number of Fibonacci spheres around a protein structure using the *fibo* program. The spheres are used in subsequent pathway analysis steps to sample spatial positions around the protein.

   Usage

```
${SILCSBIODIR}/silcs-pathway/1_cal_fibo prot=<protein PDB file>␣
↪[numfibo=<number>] [outputdir=<directory>] [cleanup=<true/
↪false>] [batch=<true/false>]
```

Required parameters

- `prot`: Full path to the protein PDB file.

Optional parameters

- `numfibo`: Number of Fibonacci spheres to generate (default: 500).
- `outputdir`: Output directory for results (default: 4_pathway).
- `cleanup`: Whether to clean up intermediate files (default: true).
- `batch`: Submit job to queuing system (default: false).

Output

- Fibonacci sphere coordinates and log file in the specified output directory.
- If successful, a message confirming completion is displayed.
- If incomplete, a warning is shown with the number of spheres generated.

Notes

- The script checks for valid input files and parameters.
- The script checks for valid input files and parameters.
- If the output directory exists, you can choose to overwrite, backup, or cancel.
- Errors and warnings are printed for missing files, invalid parameters, or failed sphere generation.

2. **A* Search Pathway Analysis**

   This step performs pathway analysis using the A* search algorithm to find optimal pathways for ligand binding or unbinding. It utilizes the generated Fibonacci spheres and the protein structure, starting from a defined binding site center.

   Usage

```
${SILCSBIODIR}/silcs-pathway/2_astar prot=<protein PDB file>␣
↪center="x,y,z" [outputdir=<directory>] [mapsdir=<FragMaps␣
↪directory>] [cleanup=<true/false>] [batch=<true/false>]
```

   Required parameters

   - `prot`: Path to the protein PDB file.

- `center`: Coordinates (x,y,z) of the binding site center (e.g., center="20.1,33,-2") (default: center.csv).

Optional parameters

- `outputdir`: Output directory for results (default: 4_pathway).
- `mapsdir`: Directory containing FragMaps (default: maps).
- `cleanup`: Whether to clean up intermediate files (default: true).
- `batch`: Submit job to queuing system (default: false).

Output

- Pathway analysis results, including pathway coordinates and PDB files, in the specified output directory.
- Log files for each search and summary of successful/failed points.
- If successful, a message confirming completion is displayed.
- If incomplete, warnings and logs are provided for debugging.

Notes

- The script checks for valid input files, parameters, and required directories.
- The output directory must contain results from step 1 (fibo_data.txt).
- Errors and warnings are printed for missing files, invalid parameters, or failed A* search.
- Failed pathway points are handled and logged for further analysis.

3. **Truncated Pathway Analysis**

This step processes the pathway points generated from the A* search and removes those that are outside the protein structure, resulting in a truncated pathway that is physically relevant for further analysis.

Usage

```
${SILCSBIODIR}/silcs-pathway/3_truncate_path prot=<protein PDB␣
↪file> [outputdir=<directory>] [batch=<true/false>] [cleanup=
↪<true/false>]
```

Required parameters

- `prot`: Path to the protein PDB file.

Optional parameters

- `outputdir`: Output directory for results (default: 4_pathway).
- `batch`: Submit job to queuing system (default: false).

- `cleanup`: Whether to clean up intermediate files (default: true).

Output

- Truncated pathway coordinates and PDB files in the specified output directory (coors/ and pathpdbs/).
- Log file indicating success or any issues encountered (truncate_<protein>.log).
- If successful, a message confirming completion is displayed.
- If incomplete, warnings and logs are provided for debugging.

Notes

- The script checks for valid input files, parameters, and required directories.
- The output directory must contain successful results from step 2 (astar_<protein>.log with "success").
- Errors and warnings are printed for missing files, invalid parameters, or failed truncation.
- Pathway points outside the protein are removed to ensure physical relevance for downstream analysis.

4. **Distance Matrix Calculation**

This step calculates the distance matrix between all truncated pathway points, providing a quantitative basis for further pathway analysis and clustering.

Usage

```
${SILCSBIODIR}/silcs-pathway/4_cal_dmatrix prot=<protein PDB␣
↪file> [outputdir=<directory>] [batch=<true/false>] [cleanup=
↪<true/false>]
```

Required parameters

- `prot`: Path to the protein PDB file.

Optional parameters

- `outputdir`: Output directory for results (default: 4_pathway).
- `batch`: Submit job to queuing system (default: false).
- `cleanup`: Whether to clean up intermediate files (default: true).

Output

- Distance matrix file (distmat.dat) in the specified output directory.
- Log file indicating success or any issues encountered (dmatrix_<protein>.log).
- If successful, a message confirming completion is displayed.

- If incomplete, warnings and logs are provided for debugging.

Notes

- The script checks for valid input files, parameters, and required directories.
- The output directory must contain successful results from step 3 (truncate_<protein>.log with "success").
- Errors and warnings are printed for missing files, invalid parameters, or failed distance matrix calculation.
- The distance matrix is used for downstream pathway clustering and analysis.

5. **Clustering Pathway Points**

This step clusters the truncated pathway points using the cluster program and removes duplicate PDBs, resulting in a set of unique pathway conformations for further analysis.

Usage

```
${SILCSBIODIR}/silcs-pathway/5_cluster prot=<protein PDB file>␣
↪[cutoff=<cutoff value>] [outputdir=<directory>] [batch=<true/
↪false>] [cleanup=<true/false>]
```

Required parameters

- `prot`: Path to the protein PDB file.

Optional parameters

- `cutoff`: Clustering cutoff value in Angstroms (default: 20.0).
- `outputdir`: Output directory for results (default: 4_pathway).
- `batch`: Submit job to queuing system (default: false).
- `cleanup`: Whether to clean up intermediate files (default: true).

Output

- Clustered and unique pathway PDB files in the specified output directory (pathunipdbs/).
- Clustering results file (cl.result) and log file (cluster_<protein>.log).
- If successful, a message confirming completion is displayed.
- If incomplete, warnings and logs are provided for debugging.

Notes

- The script checks for valid input files, parameters, and required directories.
- The output directory must contain successful results from step 4 (dmatrix_<protein>.log with "success" and distmat.dat).

- Errors and warnings are printed for missing files, invalid parameters, or failed clustering.
- Duplicate PDBs are removed to ensure a unique set of pathway conformations for downstream analysis.

6. **Visualization Script Generation**

   This step generates VMD and PyMOL scripts to visualize the clustered pathway conformations, enabling graphical inspection and presentation of the results.

   Usage

```
${SILCSBIODIR}/silcs-pathway/6_gen_visuals prot=<protein PDB␣
↪file> [clsize=<cluster size cutoff>] [outputdir=<directory>]␣
↪[batch=<true/false>] [cleanup=<true/false>]
```

   Required parameters

   - `prot`: Path to the protein PDB file.

   Optional parameters

   - `clsize`: Minimum number of members in a cluster to be included in the visualization (default: 2).
   - `outputdir`: Output directory for results (default: 4_pathway).
   - `batch`: Submit job to queuing system (default: false).
   - `cleanup`: Whether to clean up intermediate files (default: true).

   Output

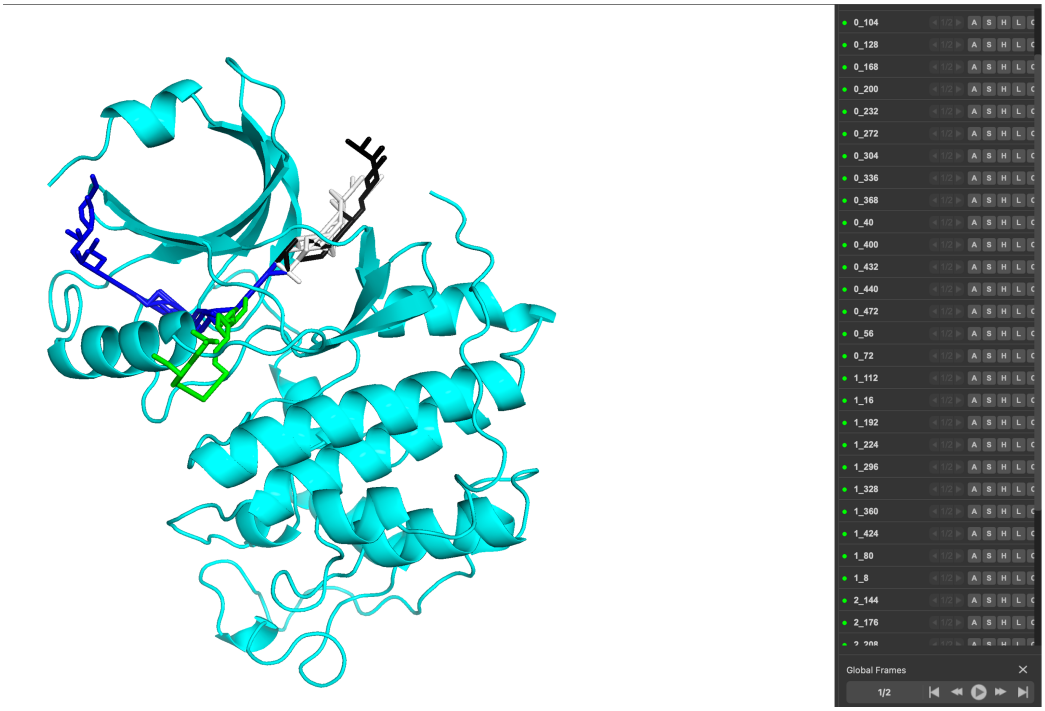
   - VMD script (pathcluster-cutoff-<clsize>.vmd) and PyMOL script (pathcluster-cutoff-<clsize>.pml) in the specified output directory.
   - Log messages indicating successful script generation.
   - If successful, a message confirming completion is displayed.
   - If incomplete, warnings and logs are provided for debugging.

   Notes

   - The script checks for valid input files, parameters, and required directories.
   - The output directory must contain successful results from step 5 (cluster_<protein>.log with "success").
   - Errors and warnings are printed for missing files, invalid parameters, or failed script generation.
   - Both VMD and PyMOL scripts are generated for flexible visualization options.

- The script names the clusters based on the specific cluster and pathway name with clusterid_pathwayid.pdb for easy identification.

An example of the visualization in pymol can be seen below:



8.8 SILCS-Kinetics

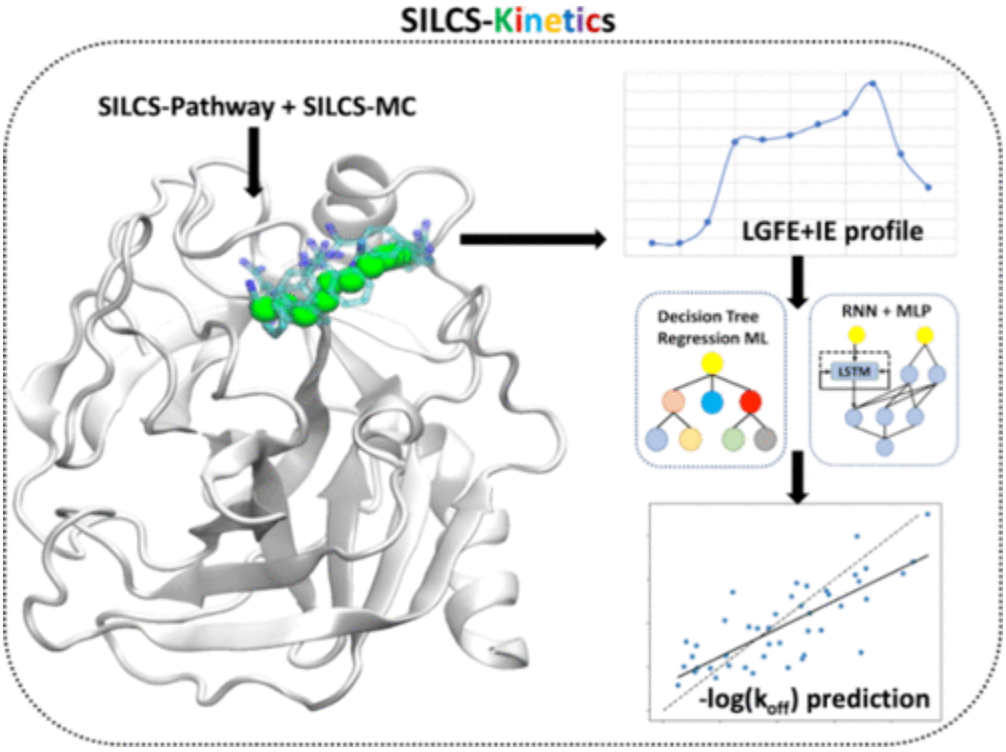
8.8.1 Background

SILCS-Kinetics is a computational workflow designed to efficiently predict ligand dissociation kinetics, specifically the off rate (koff), which is often more relevant to drug efficacy than binding affinity alone. Traditional approaches rely on enhanced-sampling molecular dynamics (MD) simulations, which are computationally intensive and less practical for evaluating large ligand libraries in drug design.

To address this, SILCS-Kinetics combines physics-based and machine learning (ML) methodologies. The workflow uses the Site Identification by Ligand Competitive Saturation (SILCS) method to enumerate potential ligand dissociation pathways and calculate free-energy profiles along those pathways. These profiles, together with molecular properties, serve as features for training ML models—including tree-based and neural network approaches—to predict koff values.

The protocol has been developed and validated on a diverse set of 329 ligands across 13 proteins, demonstrating the robustness and efficiency of the ML workflow built upon SILCS-derived free-energy profiles. SILCS-Kinetics provides a powerful and scalable tool for drug design, enabling rapid quantitative estimates of ligand dissociation kinetics and atomic or functional group contribu-

tions to unbinding events. Additional details on the development and validation of SILCS-Kinetics can be found in the original publication [25].



8.8.2 SILCS-Kinetics Using the SilcsBio CLI

1. **Cluster Generation and SILCS-MC Slab Preparation**

   This step prepares and runs the cluster-generation phase of the SILCS-Kinetics workflow for a given protein structure. It identifies ligand trajectory clusters using a specified cutoff, then creates and organizes SILCS-MC input files for every cluster–ligand pair.

   Usage

   ```
   ${SILCSBIODIR}/silcs-kinetics/1_gen_silcsmc_slab prot=
   ↪<protein PDB file>
   ```

   Required parameters

   - `prot`: Path to the protein PDB file.

   Optional parameters

   - `outputdir`: Output directory for results (default: 5_kinetics).
   - `pathwaydir`: Path to the silcs-pathway output directory (default: 4_pathway).

- `paramsfile`: Path to SILCS-MC parameters template file.
- `cutoff`: Minimum number of members in a cluster to be included in kinetics runs (minimum: 2) (default: 2).

Output

- SILCS-MC input files for each cluster–ligand pair in pathclusters/.
- Log files and cluster information.

Notes

- The script checks for valid input files and parameters.
- Clusters with fewer members than the cutoff are excluded.

2. **SILCS-MC Job Submission**

   This step submits SILCS-MC simulations for all cluster–ligand pairs identified in the previous step.

   Usage

   ::

   ${SILCSBIODIR}/silcs-kinetics/2_submit_jobs prot=<protein PDB file> ligdir=<ligand directory> ${SILCSBIODIR}/silcs-kinetics/2_submit_jobs prot=<protein PDB file> ligdir=<ligand directory>

   Required parameters

   - `prot`: Path to the protein PDB file.
   - `ligdir`: Directory containing ligand mol2/sdf files.

   Optional parameters

   - `outputdir`: Output directory for results (default: 5_kinetics).
   - `sdfile`: SD file to overwrite ligdir.

   Output

   - SILCS-MC simulation outputs for each cluster–ligand pair.
   - Log files and job status information.

   Notes

   - The script checks for valid input files and parameters.
   - Output directory must be prepared by step 1.

3. **Energy Extraction and Barrier Calculation**

This step processes the output from SILCS-MC simulations to extract energy terms and calculate barriers for each ligand–protein cluster. It also prepares data files for machine learning predictions.

Usage

::

    ${SILCSBIODIR}/silcs-kinetics/3_extract_energy prot=<protein PDB file>
    ${SILCSBIODIR}/silcs-kinetics/3_extract_energy prot=<protein PDB file>

Required parameters

- `prot`: Path to the protein PDB file.

Optional parameters

- `outputdir`: Output directory for results (default: 5_kinetics).

Output

- Extracted energy terms and barrier calculations for each cluster–ligand pair.
- Data files for ML predictions.

Notes

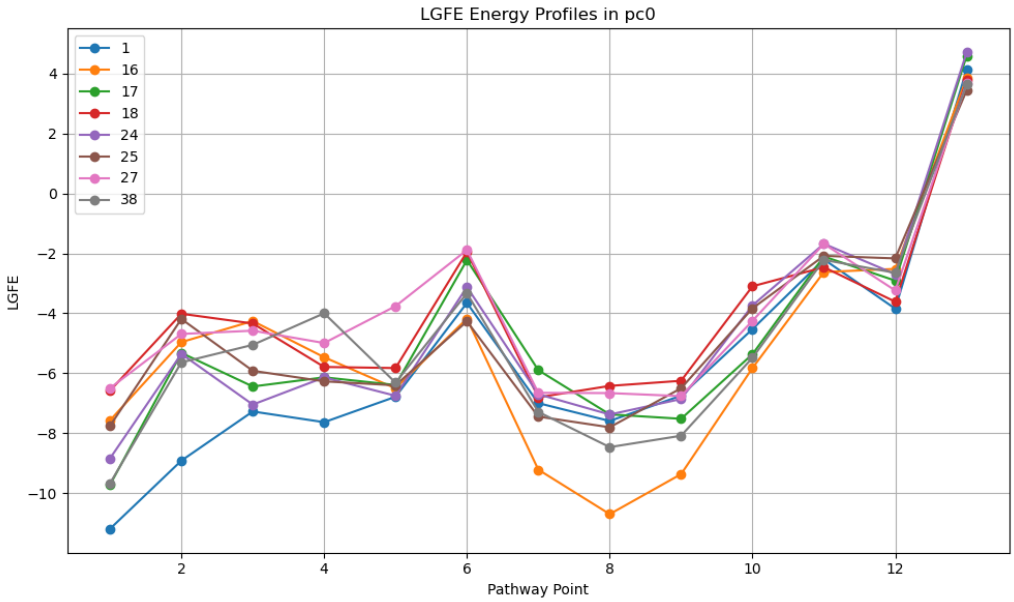
- The script checks for valid input files and parameters.
- Output directory must contain successful SILCS-MC results.
- Adds script extract_pathway_lgfe.py to outputdir which given a pc# will extract the ligand grid free energy (LGFE) along the pathway for that cluster.

Usage

::

    ${SILCSBIODIR}/silcs-kinetics/4_ml_inference prot=<protein PDB file> python extract_pathway_lgfe.py –pc N [–ligand ligand1 ligand2 . . .] [–save_plot true/false] [–show_plot true/false]

Example



4. **Machine Learning Inference**

   This step performs the final ML inference, extracting features from SILCS-MC outputs and applying trained models to predict –log(koff) values for each ligand–protein cluster pair.

   Usage

   ::

      ${SILCSBIODIR}/silcs-kinetics/4_ml_inference prot=<protein PDB file>

   Required parameters

   - `prot`: Path to the protein PDB file.

   Optional parameters

   - `outputdir`: Output directory for results (default: 5_kinetics).
   - `ml_model`: ML model type to use for inference (RF, RNN, or both; default: RF).
   - `python`: Path to Python executable.

   Output

   - Predicted –log(koff) values for each ligand–protein cluster pair.
   - ML inference log files and results.

   Notes

   - The script checks for valid input files, parameters, and model type.
   - Output directory must contain successful energy extraction results.

8.8.3 Training the ML Models using the CLI

Scripts are provided to train the ML models used in the SILCS-Kinetics workflow. These scripts can be run from the command line and require specific input data for training. These scripts are located in the utils directory:

```
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/{tree/rnnmlp}/
↪train/training/
```

The structured input data for each model varies slightly but both include the silcs-kinetics energy terms, physical properties and experimental data. For the RF model, the input data is a CSV file with the following columns:

1. ligand_id
2. Sum of the LGFE barriers
3. Sum of the IE differences
4. LGFESMS
5. MW
6. Number of rotatable bonds (NRot)
7. Experimental –log(koff)

Numbers 1-4 are dervived from SILCS-Kinetics and can be found in the ml.bar file generated in step 3 of the workflow. Number 5-6 are physical properties which can be calculated independently or using an included python script.

```
python $SILCSBIODIR/utils/python/silcs-kinetics/sdf2smi.py
python $SILCSBIODIR/utils/python/silcs-kinetics/physchem_calc.py
```

Alternatively, you can run your data through step 4 of the workflow which will run the above scripts and generates all.prop file which contains the required properties for all excipients or in the 4_ml/tree directory is a file ml-rf.csv which contains the properly structured input for all ligands.

Training for the RNN-MLP model requires a different input format. The train data is split into two files and a directory. In order to generate the required input to generate this input you should run the RNN-MLP model described above. After running the RNN-MLP model, the required input file will appear in the 5_kinetics folder. In the run directory (where 5_kinetics/ directory is located) you should add a file neg-logkoff.exp which should have two columns excipient name and experimental -log(koff) value separated by a tab. A script called prep_rnn_input.sh can be used to prepare the input for a specific protein for the RNN-MLP model. This script is located at:

```
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/rnnmlp/train/
↪training/prep_rnn_input.sh
```

After preparing the files ml-rnn-input.csv for each protein in the test/train set all these files should be concatented into path.train, path.test, and path.val for your training, testing and validation set. A script called rnnmlp_preprocessing.sh is provided to take the path files generated and normalize them to prepare for training. This script is located at:

```
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/rnnmlp/train/
↪training/rnnmlp_preprocessing.sh
```

Note this script will need run for separately for the test, train and validation sets. After running this script you will have the properly formatted input for training the RNN-MLP model. These input files should then be moved to the train/data directory located at:

```
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/rnnmlp/train/data/
```

The scripts for training the RNN-MLP model are located in the following directory:

```
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/rnnmlp/train/
↪training/rnnmlp_train.py
${SILCSBIODIR}/utils/python/silcs-kinetics/ml_inference/rnnmlp/train/
↪training/rnnmlp_tuner.py
```

The tuner script will retune hyperparameters and train a new model, while the train script will train a new model using teh default hyperparameters from original model tuning.

Instructions for training the models can be found in the README files in the tree/rnnmlp directories within the $SILCSBIODIR/utils/python/silcs-kinetics/ml_inference/ directory.

CHAPTER

NINE

BIOLOGICS SUITE

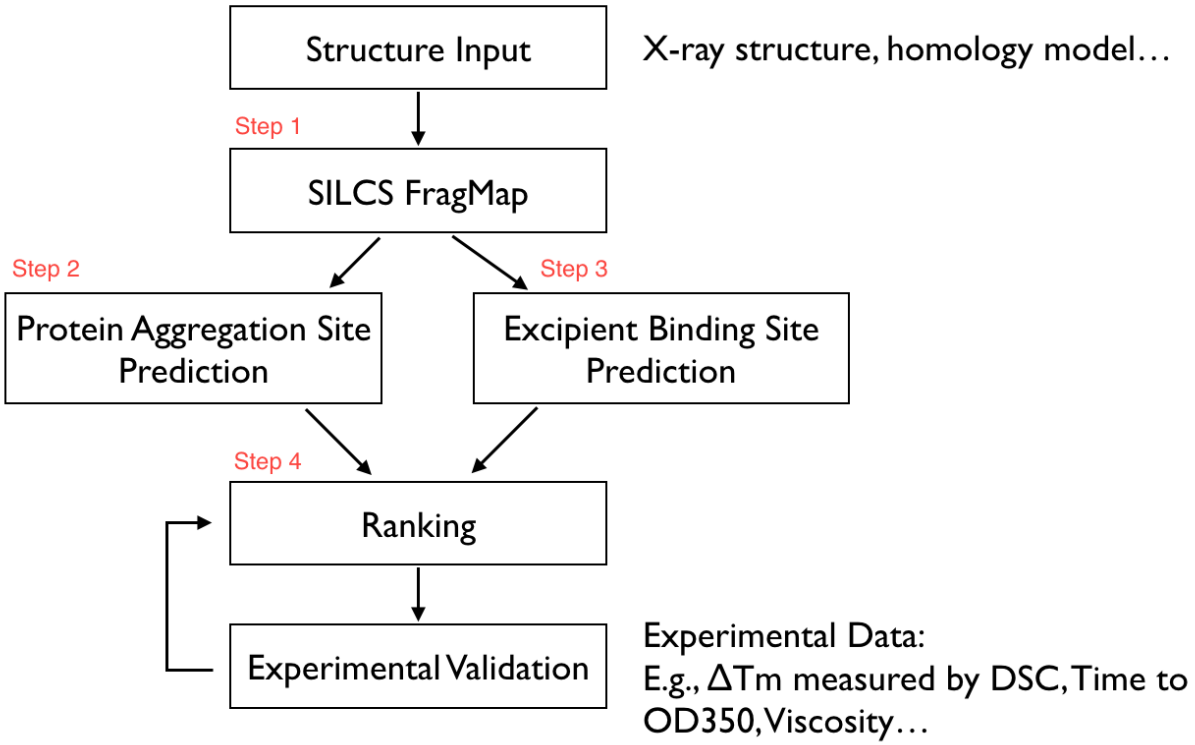
9.1 Background

Biomacromolecular therapeutics, commonly called "biologics," are typically protein molecules that have been developed to selectively interact with a therapeutic target. Common examples of biologics are therapeutic antibodies. Biologics must be carefully formulated to maximize their stability, so as to ensure both efficacy and safety. Important determinants of stability are protein aggregation and protein denaturation. Toward maximizing stability, biologics can be formulated with excipients, which can help minimize aggregation and denaturation of the biologic in a solution formulation.

SILCS-Biologics applies the patented Site Identification by Ligand Competitive Saturation (SILCS) platform technology to the rational selection of excipients for biologics formulations. The first goal is to minimize protein–protein interactions between multiple molecules of the protein, all in the native state. This reduces the likelihood of aggregation of the protein in its native state. The second goal is to minimize denaturation by stabilizing the native (folded) state of the protein. This is accomplished by screening for excipients that can efficiently bind to the protein native state. Such binding drives the equilibrium toward the native state and away from denatured states. Achieving the second goal also contributes to minimizing aggregation, as denatured states of the protein can be aggregation-prone.

The SILCS-Biologics workflow consists of four steps:

1. SILCS simulations
2. Protein–protein interaction screening
3. Protein-excipient hotspots screening
4. analysis and ranking.



SILCS-Biologics is actualized as a unified tool, available through both the command line and the SilcsBio Graphical User Interface, that integrates all four steps. As such, SILCS-Biologics automatically takes care of preparing and running computing jobs that entail SILCS to generate FragMaps, SILCS-PPI to predict protein–protein interactions (PPI) that can drive aggregation, and SILCS-Hotspots to determine excipient binding sites. Please see *SILCS Simulations* and *SILCS-Hotspots* for additional details. SILCS-PPI works by aligning FragMaps from one protein with the functional groups on the surface of another protein to predict the likelihood of a PPI between the two proteins [26]. Please see *Protein–Protein Interaction Maps* for additional information.

9.2 Usage

In principle, SILCS-Biologics can be applied to any protein therapeutic. In practice, because SILCS is an all-atom explicit-solvent molecular dynamics methodology, the SILCS simulations underpinning SILCS-Biologics can become computationally prohibitive for large proteins, like full-sequence antibodies or non-antibody protein therapeutics of large size. To enable application of SILCS-Biologics to larger proteins, you may split your full protein into two or three domains. SILCS-Biologics will automatically take care of managing the separate SILCS simulations for each domain and the subsequent SILCS-Hotspots and SILCS-PPI analyses, as well as collating the separate data into a single report for the full-length protein. The usage of SILCS-Biologics with the *SilcsBio GUI* and the *CLI* is described below. It is strongly recommended that users read and understand the descriptions of the use cases before using either the command line or the SilcsBio GUI to run

SILCS-Biologics.

9.2.1 SILCS-Biologics Using the SilcsBio GUI

Please see *SILCS-Biologics Using the GUI* in the *Graphical User Interface (GUI) Quickstart* for instructions on running SILCS-Biologics from the SilcsBio GUI.

9.2.2 SILCS-Biologics Using the CLI

The SILCS-Biologics workflow can be run using the following command:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics step=<1*2*3*4*> prot1=
↪<protein PDB>
```

Required parameter:

- Tasks to be completed:

  ```
  step=<1*2*3*4*>
  ```

  All tasks can be submitted at once with a single command with `step=1*2*3*4*`. To submit specific tasks or subtasks one step at a time, specify the task by entering only the desired task (e.g., `step=1*` to submit only the first task). The tasks and subtasks are listed below:

  - `step=1*2*3*4*` or `step=1abc2ab3ab4ab`: Run all steps
  - `step=1*`: Run SILCS simulations and generate FragMaps
    - `step=1a`: set up SILCS simulations
    - `step=1b`: run SILCS simulations
    - `step=1c`: generate FragMaps
  - `step=2*`: Run SILCS-PPI (predict protein aggregation sites)
    - `step=2a`: Run protein–protein docking simulations with SILCS-PPI
    - `step=2b`: Collect SILCS-PPI data
  - `step=3*`: Run SILCS-Hotspots (predict excipient binding sites)
    - `step=3a`: Dock excipients onto the target protein with SILCS-Hotspots
    - `step=3b`: Cluster docked poses from SILCS-Hotspots
  - `step=4*`: Generate SILCS-Biologics report
    - `step=4a`: Generate web report
    - `step=4b`: Generate spreadsheet report

- Path and name of protein PDB file:

```
prot1=<protein PDB>
```

Optional parameters:

- Path and name of second protein domain PDB file:

```
prot2=<protein 2 PDB>
```

The target protein may be divided into domains for computational expediency. In this case, each domain is input as a separate PDB file. For a two domain system, `prot1` will correspond to one domain, and `prot2` will correspond to the remaining domain. For a three domain system, `prot1` will correspond to the first domain, `prot2` will correspond to the second domain, and `prot3` will correspond to the third domain. If `prot2` or both `prot2` and `prot3` are used, `fullpdb` must also be specified (see below).

- Path and name of third protein domain PDB file:

```
prot3=<protein 3 PDB>
```

The target protein may be divided into domains for computational expediency. In this case, each domain is input as a separate PDB file. For a three domain system, `prot1` will correspond to the first domain, `prot2` will correspond to the second domain, and `prot3` will correspond to the third domain. If `prot2` and `prot3` are used, `fullpdb` must also be specified (see below).

- Path and name of full-length protein PDB file (required if more than one protein is input):

```
fullpdb=<full-length protein PDB>
```

If `prot2` or both `prot2` and `prot3` are used, then the full-length protein PDB file must also be specified with `fullpdb`.

Note: If you do not have the full-length structure of the target protein, but only have the structures of individual domains, then the full-length protein structure should be constructed using other molecular modeling tools (such as homology modeling software). The modeled full-length structure can then be used as the "Reference PDB File".

Alternatively, the positions of the individual domains in the context of the full-length protein can be estimated through a simple structural alignment to a known full-length homologous crystal structure, and the resulting superimposed structure of the individual domains can be saved and used as the "Reference PDB File".

Warning: Dividing the target protein into domains entails independent SILCS simulations for each of the domains. E.g., for SILCS-Biologics runs with `prot1`, `prot2`, and `prot3` specified (3 domains) 3 × 10 SILCS simulations will be performed rather than 1 × 10 SILCS simulations. If sufficient compute nodes are available to run all SILCS simulations for all domains simultaneously, the SILCS simulations for the target protein will be completed sooner using the domain approach than running SILCS simulations of the full-length protein. However, using the domain approach may consume more CPU/GPU hours.

- Path and name of directory containing excipient Mol2/SD files:

```
molsdir=<directory of the excipient files; default=
↪$SILCSBIODIR/data/excipients/mols>
```

The excipient structure files in the `molsdir` directory may be in Mol2 or SD file format. In both formats, only one excipient per file will be read.

Tip: If you have a single SD file containing multiple excipients, you may split the file into multiple SD files with one excipient per file using the following command, where <multiple.sdf> is the SD file containing multiple excipients and <molsdir> is the directory in which all output SD files containing one excipient per file will be placed:

```
${SILCSBIODIR}/programs/format_convertor --split --inf <multiple.sdf>
--outf <molsdir>
```

- Path and name of buffer molecule Mol2/SD file:

```
buffer=<path and name of buffer molecule; default="NONE">
```

Only one buffer molecule can be specified at a time. The buffer molecule file may be in Mol2 or SD file format. In both formats, only one excipient per file will be read. The buffer molecule file can be located in a different directory than `molsdir`.

- ID for the job:

```
job_id=<id for the current job; default=$jobTime (timestamp)>
```

- Path and name of project working directory:

```
workdir=<directory of the project; default=$PWD>
```

- Name of directory for first task (step 1):

```
dir1=<directory for step 1; defaul=$workdir/1_fragmap.$job_
↪id>
```

- Name of directory for second task (step 2):

```
dir2=<directory for step 2; defaul=$workdir/2_ppi.$job_id>
```

- Name of directory for third task (step 3):

```
dir1=<directory for step 3; defaul=$workdir/3_excipients.
↪$job_id>
```

- Name of directory for fourth task (step 4):

```
dir4=<directory for step 4; defaul=$workdir/4_report.$job_id>
```

9.2.3 Viewing Step 4 Reports

Once all four steps have completed, reports will be ready for viewing and analysis.

Tip: If you ran your SILCS-Biologics workflow using the SilcsBio GUI, these reports will also be available for viewing via the SilcsBio GUI. Please refer to *SILCS-Biologics Using the GUI* for more information.

Tip: The data can also be used to design experiments for a biologic in development using the following approach:

1. Identify excipients with high frequency of binding at sites with select PPIP cutoffs.
2. Select excipients with low frequency of binding at sites with select PPIP cutoffs as negative controls.
3. Design experiments to test the effects of these excipients on the stability of the biologic, and compare the results for excipients

---

Viewing through a Spreadsheet Application

If you wish to open these data in a spreadsheet application, you can find them in `$WORKDIR/4_report.$job_id/`, where `$WORKDIR` is the top-level directory containing all the `silcs-biologics` workflow outputs. By default, `$job_id` is set based on the system date and time, and `$WORKDIR` is set to the directory in which the `silcs-biologics` command was executed. You can override these defaults with the options `job_id=` and `workdir=` when executing `silcs-biologics`.

Tip: See *SILCS-Biologics Directory Structure* for a complete description of how `silcs-biologics` organizes and names the directories and files that it creates.

`$WORKDIR/4_report.$job_id/report_all.xlsx` has a tab named "Ranking" that contains a number of properties on a per-excipient basis:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Excipient	# Binding Site (LGFE < -1)	# Binding Site (LE < -0.25)	# Binding Site (RAA < 1000)	# Binding Site (PPIP > 0.10)	# Binding Site (LE < -0.25 and PPIP > 0.10)	Ave LGFE	Ave LE	Lowest LGFE	Sum(PPIP) /# Sites	# Binding Site ∩ Buffer	# Binding Site ¢ Buffer	# Binding Site (RAA_BUFFER < 1)
2	histidine	79	40	24	6	2	-2.79	-0.25	-5.87	0.04	0	79	1
3	proline	103	70	41	6	2	-2.44	-0.3	-5.26	0.04	0	103	0
4	sorbitol	64	53	18	3	3	-4.52	-0.38	-8.17	0.04	0	64	20
5	aspartate	67	40	34	4	4	-2.56	-0.28	-5.169	0.04	0	67	0
6	glutamate	71	45	23	4	3	-2.92	-0.29	-5.876	0.04	0	71	1
7	threonine	98	84	43	4	3	-2.74	-0.34	-5.49	0.04	0	98	1
8	valine	101	72	37	5	3	-2.55	-0.32	-5.38	0.04	0	101	1
9	alanine	115	85	69	4	4	-1.93	-0.32	-4.32	0.04	0	115	0
10	succinate	52	50	35	3	2	-2.93	-0.37	-5.17	0.04	0	52	0
11	glycine	105	94	105	3	3	-1.83	-0.37	-3.71	0.03	0	105	0
12	phosphate	117	105	113	8	8	-1.8	-0.36	-3.9	0.04	0	117	0
13	arginine	55	31	7	3	0	-3.34	-0.28	-6.94	0.04	0	55	2
14	sucrose	42	23	6	1	1	-6.06	-0.26	-10.446	0.03	0	42	26
15	mannitol	56	51	20	4	3	-4.59	-0.38	-7.88	0.04	0	56	16
16	glucose	79	66	23	4	2	-4.21	-0.35	-7.48	0.04	0	79	16
17	citrate	52	41	31	2	1	-4.1	-0.32	-6.277	0.04	0	52	10
18	lactate	106	106	106	7	7	-2.35	-0.39	-4.18	0.04	0	106	0
19	lysine	60	50	3	3	2	-3.39	-0.34	-7.47	0.04	0	60	1
20	malate	68	66	40	3	3	-3.34	-0.37	-5.57	0.04	0	68	1
21	trehalose	42	24	8	1	1	-6.21	-0.27	-10.86	0.04	0	42	25

The contents of the columns are:

A: The excipient.

B: The number of binding sites where the excipient binds with an Ligand Grid Free Energy (LGFE) < -1 kcal/mol. LGFE approximates the binding affinity.

C: The number of binding sites where the excipient Ligand Efficiency (LE) < -0.25 kcal/mol. LE provides a metric of the binding affinity normalized for molecular size. LE = LGFE / N_heavy_atom, where N_heavy_atom is the number of non-hydrogen atoms in the excipient.

D: The number of binding sites found with a Relative Affinity Analysis metric (RAA) < 1000. RAA

indicates the number of "strong" binding sites accessible to each molecule. The RAA is a ratio of dissociation constants (Kd's), where the numerator is the Kd for the excipient at a given pocket and the denominator is the Kd for the best-scoring (highest affinity) binding pose for that excipient across the entire protein. Kd is computed from LGFE, where LGFE is taken to be the free energy.

E: The number of binding sites with a sum of PPI preference for residues in the binding site (PPIP) > 0.10. A high PPIP value for a binding site suggests that site is more likely to contribute to a PPI.

F: The number of binding sites having both LE < -0.25 kcal/mol and PPIP > 0.10.

G: The average LGFE for the excipient.

H: The average LE for the excipient.

I: The LGFE for the best-scoring (highest affinity) pose of the excipient.

J: Sum(PPIP)/# sites is the sum of the PPIP values for all of the binding sites in column B divided by the value of column B. This gives an average PPIP value computed accross the excipient-favored binding sites.

K: The number of binding sites that overlap with buffer binding sites. If no buffer was specified, this number will be zero. If all excipient binding sites are also capable of binding buffer, this number will be equal to the value in column B.

L: The number of excipient binding sites that are not also buffer binding sites. If no buffer was specified, this number will be equal to the value in column B.

M: The number of binding sites where the excipient has a higher binding affinity (i.e, lower Kd or, equivalently, more negative LGFE) than the buffer at that site. In otherwords, the number of binding sites where this excipient will out-compete buffer for binding. If no buffer was specified, this computation is done relative to a hypotheical buffer with an LGFE of -4 kcal/mol.

`$WORKDIR/4_report.$job_id/report_output.xlsx` contains fractional occupancy data for excipient binding sites. Individual sheets or tabs in this file correspond to separate input formulations. Information includes the excipients and buffer that bind at each site identified for that formulation along with selected SILCS metrics associated with the occupancy calculations and PPI interactions. The top row identifies the buffer and excipients used in that particular formulation along with their concentrations.

With regard to occupancies,

- Kd: dissociation constant, such that LGFE=R*T*ln(Kd)
- [L]: concentration of the ligand/excipient, with unit Wt/V (g/mL) % => mM
- Occupancy = [L]/([L] + Kd) and has range (0, 1). A high occupancy value (close to 1) means that excipient molecule has a high probability of binding at the site. NOTE: Occupany values relative to buffer are also accessible.

Below is an example of fractional occupancy data. The example metrics assume an occupancy cutoff of 0.8. This and other terms may be varied for reporting.

Excipient	# Binding Site (PPIP > 0.10)	# Binding Site (Occup. > 0.80)	# Binding Site (Occup. > 0.80 and PPIP > 0.10)	# Binding Site (Excipient1) (Occup. > 0.80)	# Binding Site (Excipient1) (Occup. > 0.80 and PPIP > 0.10)	# Binding Site (Excipient2) (Occup. > 0.80)	# Binding Site (Excipient2) (Occup. > 0.80 and PPIP > 0.10)	# Binding Site (Excipient3) (Occup. > 0.80)	# Binding Site (Excipient3) (Occup. > 0.80 and PPIP > 0.10)
A-Met	23	161	23	161	23	0	0	0	0
A-Pro-Met	43	180	31	137	20	43	11	0	0
A-Met-Arg	35	175	27	130	19	45	8	0	0
A-Pro-Tre-Iso	53	188	32	92	16	85	12	11	4
A-Pro-Iso	43	185	30	158	23	27	7	0	0
A-Pro-Ala	47	131	27	105	22	26	5	0	0
A-Pro-Tre-Arg	54	181	30	85	12	59	9	37	9
A-Pro-Tre-Ala	57	150	27	86	12	53	13	11	2
A-Pro-Tre	47	170	33	86	12	84	21	0	0
A-Pro-Lys-Tre	54	174	31	85	12	61	12	28	7
A-Pro	37	176	33	176	33	0	0	0	0
A-Iso-Tre	43	220	36	135	24	85	12	0	0
A-Pro-Lys	44	160	31	92	16	68	15	0	0
A-Pro-Arg-Glu	50	161	27	83	15	40	9	38	3
A-Ala	39	134	27	134	27	0	0	0	0
A-Lys	22	132	22	132	22	0	0	0	0
A-Arg	23	122	21	122	21	0	0	0	0
A-Arg-Glu	34	147	24	76	13	71	11	0	0
A-Tre	12	86	12	86	12	0	0	0	0
A-Su	15	87	15	87	15	0	0	0	0
A-NaCl	12	11	3	11	3	0	0	0	0

"# Binding Site (Excipient1),(Occup. > 0.80)" suggests the number of excipient binding sites with occupancy > 0.8. If the formulation has multiple excipients, then "excipient1" also represents the excipient with the most binding sites, n comparison with other excipient(s).

"# Binding Site (Excipient2),(Occup. > 0.80)" suggests the number of sites occupied by the excipient with 2nd most binding sites.

"# Binding Site (Excipient3),(Occup. > 0.80)" suggests the number of sites occupied by the excipient with 3rd most binding sites.

"# Binding Site (Excipient1), (Occup. > 0.80 and PPIP > 0.10)" suggests the number of excipient binding sites with high occupancy that coincide with likely PPI. If the formulation has multiple different excipients, then "excipient1" also represents the excipient with the most binding sites, in comparison with other excipient(s).

"# Binding Site (Excipient2),(Occup. > 0.80 and PPIP > 0.10)" suggests the number of second most excipient binding sites with high occupancy that coincide with likely PPI.

"# Binding Site (Excipient3),(Occup. > 0.80 and PPIP > 0.10)" suggests the number of third most excipient binding sites with high occupancy that coincide with likely PPI.

"# Binding Site (Occup. > 0.80)" is the sum of any excipient binding sites with high occupancy. Each binding site with higher occupancy will be counted once.

"# Binding Site (PPIP > 0.1)" suggests the number of all excipient binding sites that coincide with potential PPI.

"# Binding Site (Occup. > 0.80 and PPIP > 0.10)" is the sum of all excipient binding sites with high occupancy that coincide with likely PPI.

Viewing through plotting scripts

A series of post-processing scripts are included in the SILCS-Biologics workflow through the CLI. These scripts can be used to generate plots for binding predictions, and to generate a series of SILCS-Biologics results for different user-inputed formulations with experimental data.

The scripts will be located in the 4_report.job_id directory, and can be run with the following commands:

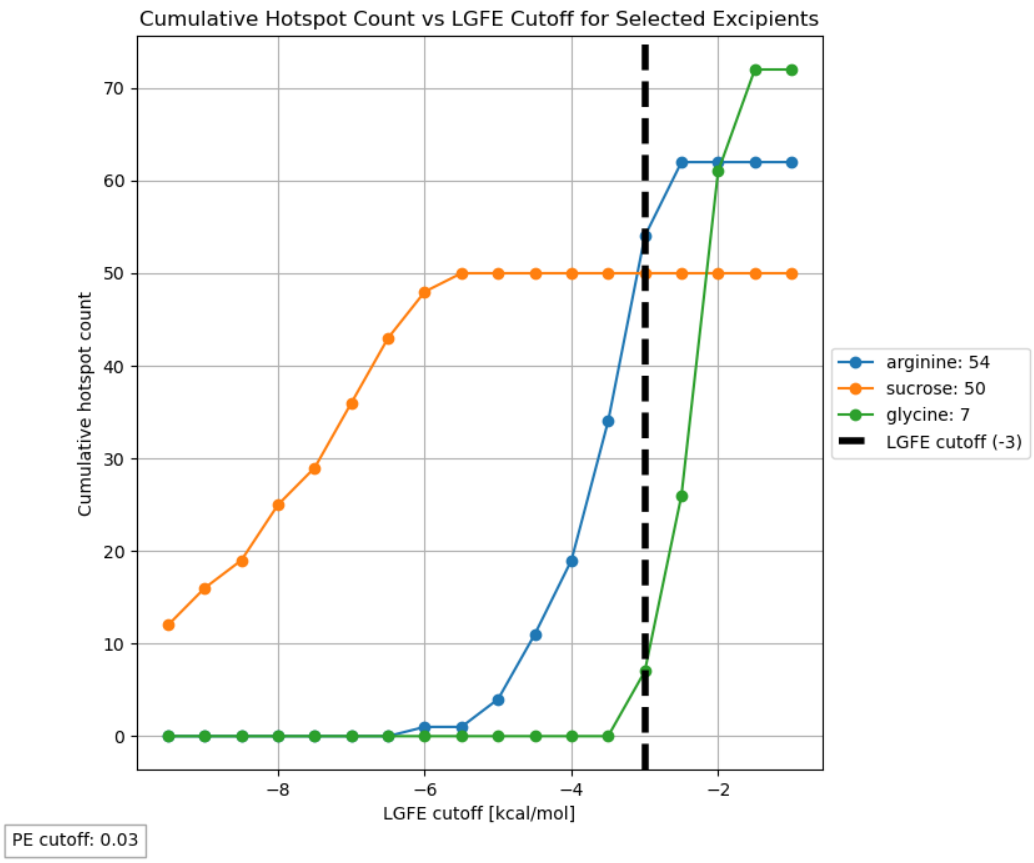
The plotting script can be used to generate plots for binding predictions:

```
python plotting_biologics_database.py
```

Parameters

- `-h --help` : show help message and exit
- `--lgfe_cutoff`: LGFE cutoff value for binidng site inclusion (default: -3)
- `--pe_cutoff`: PE cutoff value for binding site inclusion (default: 0.03)
- `--le_cutoff`: LE cutoff value for binding site inclusion (default: 0)
- `--excipients`: List of excipients to analyze (default: arginine glycine sucrose)
- `--excipient_list`: List all unique excipients exit
- `--visualization`: Display plot window in addition to saving (default: false)
- `--include_txt`: If true, output txt file containing the rank ordering for selected excipients is written (default: false).
- `--output_plot`: Output PNG filename (excipient_list_cumulative_hotspot_lgfecutoff_pecutoff.png).
- `--save_plot`: Save the plot to file (default: true). Note: If –save_plot is false, –visualization will be set to true unless you explicitly set –visualization false. If you do not want to save the plot or see the visualization, you must set both –save_plot false and –visualization false.
- `-d --default`: Run the script with default settings.

Running with the default settings generates the following plot:



The script to generate a series of SILCS-Biologics results for different user-inputed formulations with experimental data (using expt_data.csv) can be run with the following command:

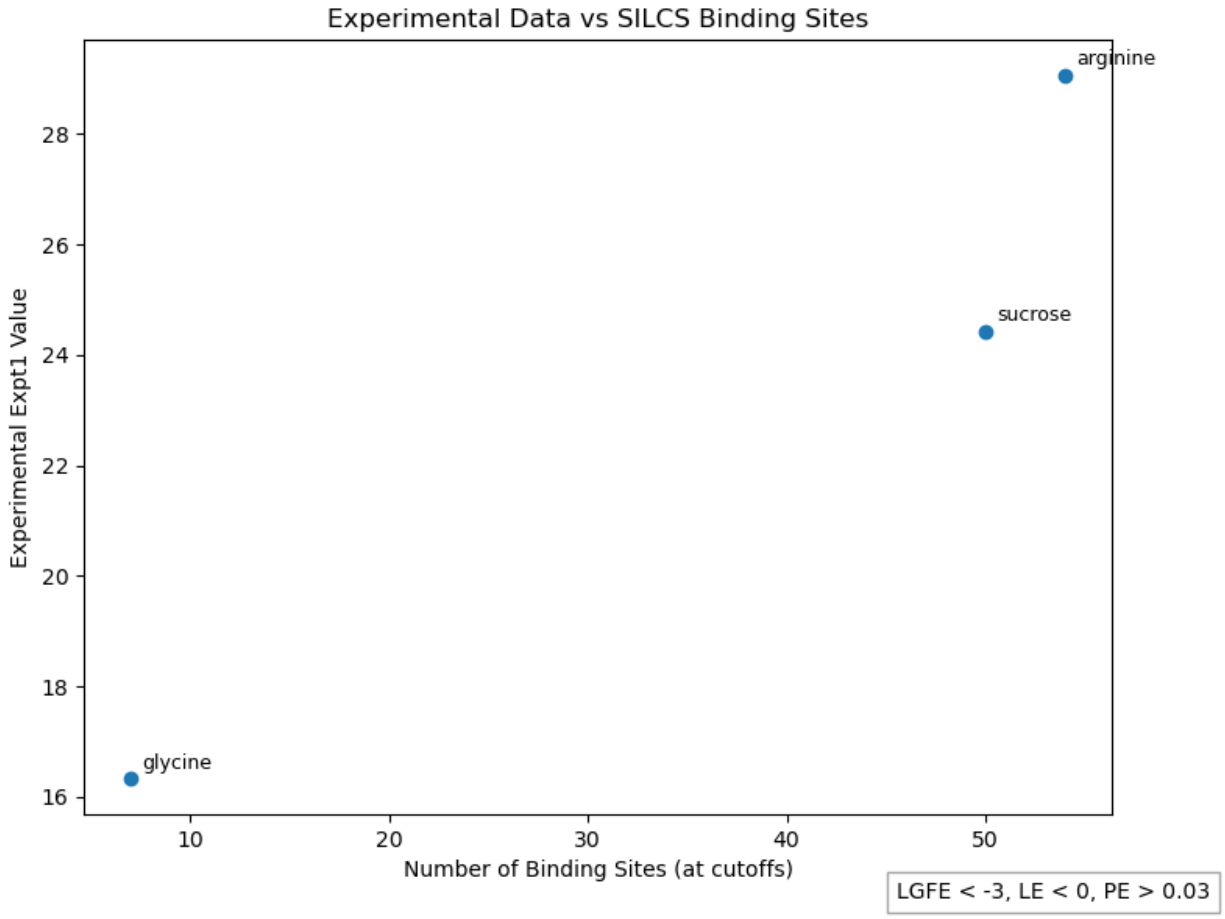
```
python plotting_biologics_rank_ordering.py
```

With no options the script prints out the following:

- `-h --help` : show help message and exit
- `--lgfe_cutoff`: LGFE cutoff value for binidng site inclusion (default: -3)
- `--pe_cutoff`: PE cutoff value for binding site inclusion (default: 0.03)
- `--le_cutoff`: LE cutoff value for binding site inclusion (default: 0)
- `--excipients`: List of excipients to analyze (default: arginine glycine sucrose)
- `--expt_excipient`: List all excipients from expt_data.csv and exit.
- `--excipient_mut`: List all unique excipients present in both report.db and expt_data.csv and exit.
- `--visualization`: Display plot window in addition to saving (default: false)
- `--output_plot`: Output PNG filename (expt_data_vs_silcs_binding_lgfecuoff_lecutoff_pecutoff.png).

- `--save_plot`: Save the plot to file (default: true). Note: If –save_plot is false, –visualization will be set to true unless you explicitly set –visualization false. If you do not want to save the plot or see the visualization, you must set both –save_plot false and –visualization false.
- `-d --default`: Run the script with default settings.

Running with the default settings generates the following plot:



In addition to the plotting scripts, a script is also included to generate a series of spreadsheet results for different user-inputed formulations with experimental data

```
occupancy_xls.py
occup.sh
```

Will produce a single xlsx file with the occupancy data for all user-inputed formulations in expt_data.csv and is run with the following command:

```
sh occup.sh
```

If you instead want to run a series of these calculations you can use the following scripts:

```
run_loop.sh
occup_loop.sh
occupancy_xls.py
```

This will produce a serires of csv files with occupancy at specificed LGFE, PE, and LE cutoffs for each user-inputed formulation in expt_data.csv which the user can control in run_loop.sh. In order to start the series of calculations, run the following command:

```
sh run_loop.sh
```

Viewing through a Web Browser

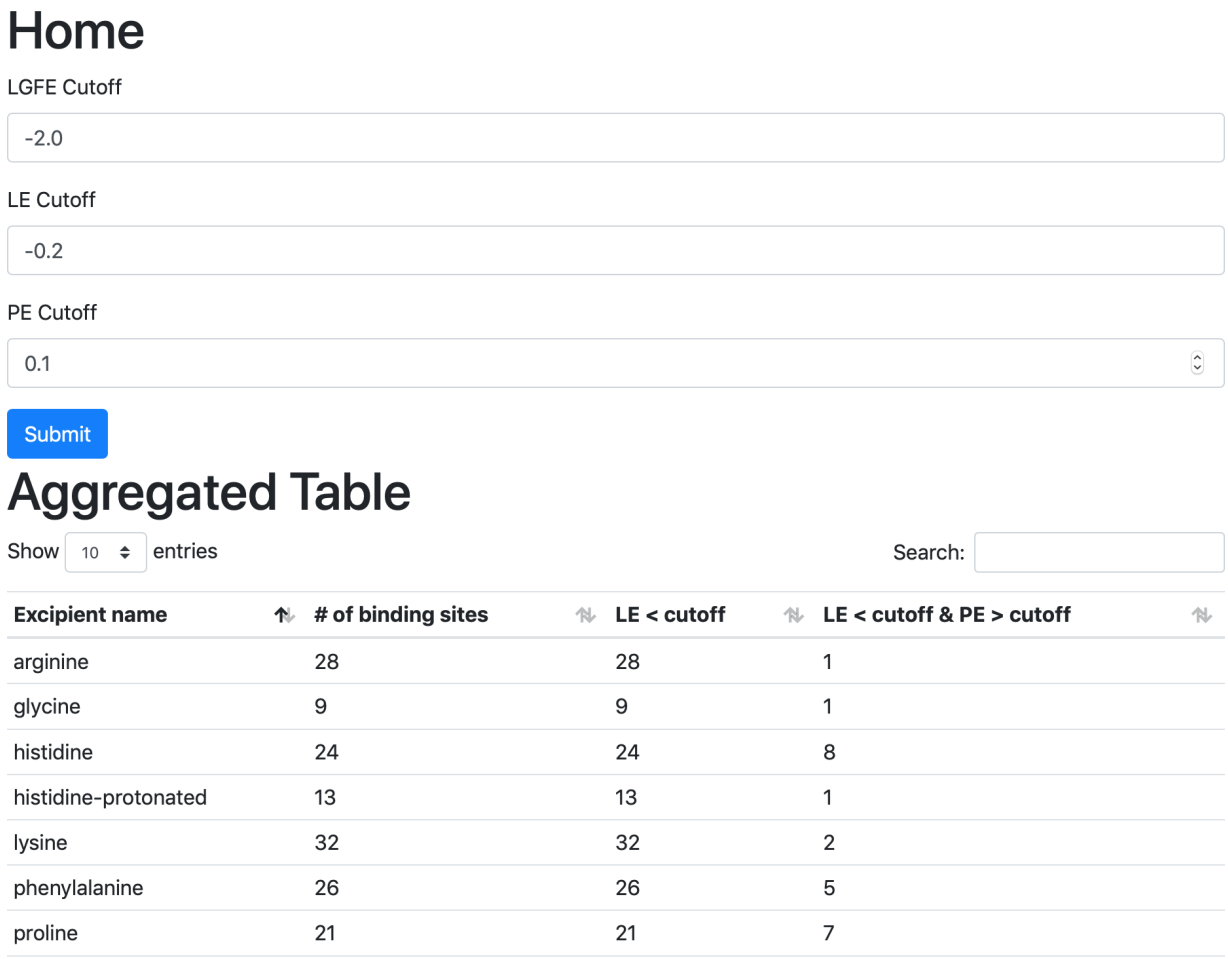
While a spreadsheet application or the SilcsBio GUI is the preferred means to view `.xlsx` files, a web view option is available for `report_all.xlsx`:

```
cd $WORKDIR/4_report.$job_id/view
sh run.sh
```

`sh run.sh` will print out a message like:

```
* Serving Flask app "main.py" (lazy loading)
* Environment: development
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 222-574-256
```

Open a web browser and go to the URL displayed in the output (e.g., http://127.0.0.1:5000 or http://<IP address>:5000).



Home

LGFE Cutoff

-2.0

LE Cutoff

-0.2

PE Cutoff

0.1

Submit

Aggregated Table

Show 10 entries Search:

Excipient name	# of binding sites	LE < cutoff	LE < cutoff & PE > cutoff
arginine	28	28	1
glycine	9	9	1
histidine	24	24	8
histidine-protonated	13	13	1
lysine	32	32	2
phenylalanine	26	26	5
proline	21	21	7

If you are running the web view from a remote server and if the server is behind a firewall, you may not able to directly access the web view. In this case, you can use a method called SSH port forwarding. This uses an SSH connection to the remote server to access the web view securely.

If you are using Linux, MacOS, or Linux subsystem on Windows as your desktop:

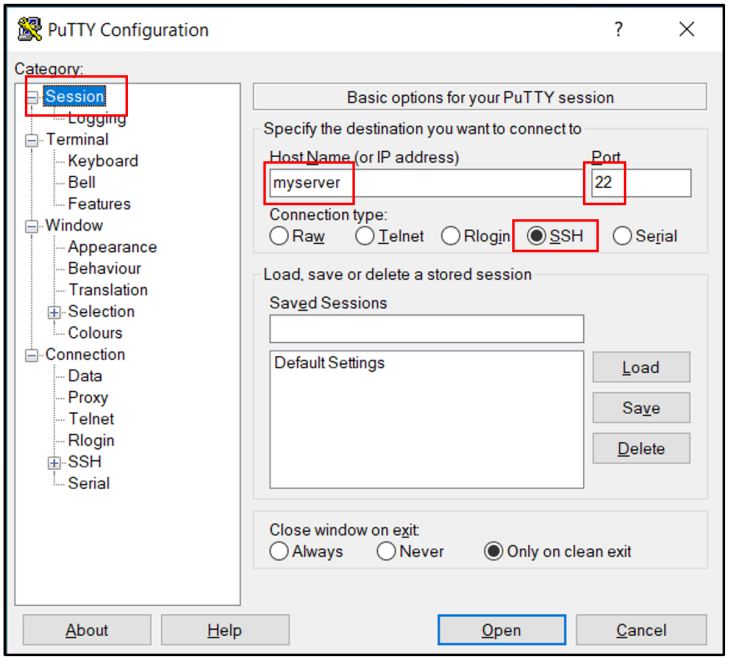
1. Open a terminal;
2. SSH to the remote server using the following example SSH command in the terminal;

```
ssh -L 5000:localhost:5000 username@servername
```

3. Open a tab in the web browser, and navigate to `localhost:5000`.

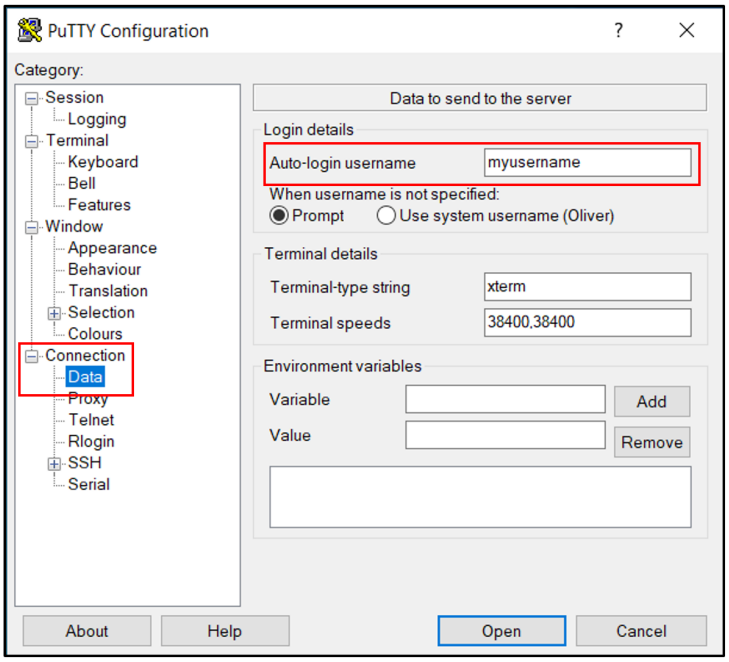
If you are using Windows as your desktop and the PuTTY application for SSH access:

1. Open PuTTY
2. Select “Session”, and
   - Type in the target host name and port (default: 22);
   - Choose SSH as the connection type (default);



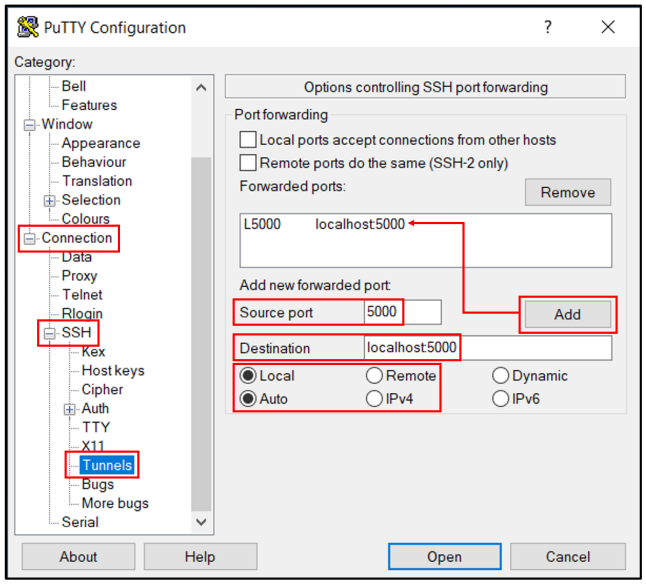
3. Select “Connection -> Data”, and

   - Type in your username as the auto-login username

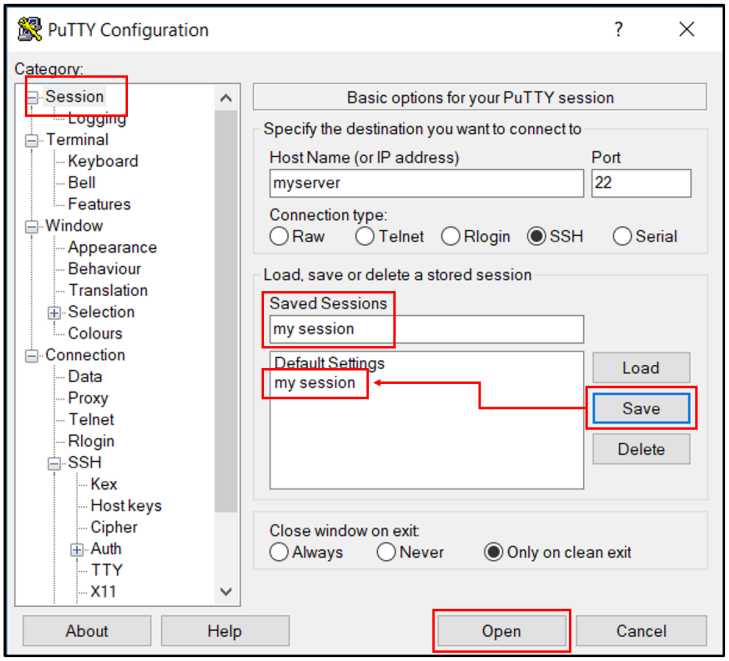


   - Select “Connection -> SSH -> Tunnels”, and
     - In “Source port”, type in `5000`;
     - In “Destination”, type in `localhost:5000`;
     - Click the “Add” button on the right side of “Source port” box;

- Turn on the *Local* and *Auto* radio buttions (default) below the "Destination" box;



4. Select "Session", and
    1. Choose a name for the session;
    2. Click the "Save" button to save the session with all your parameters;
5. Click "Open" to open a terminal to connect to your remote server.



6. Open a tab in the web browser, and navigate to `localhost:5000`.

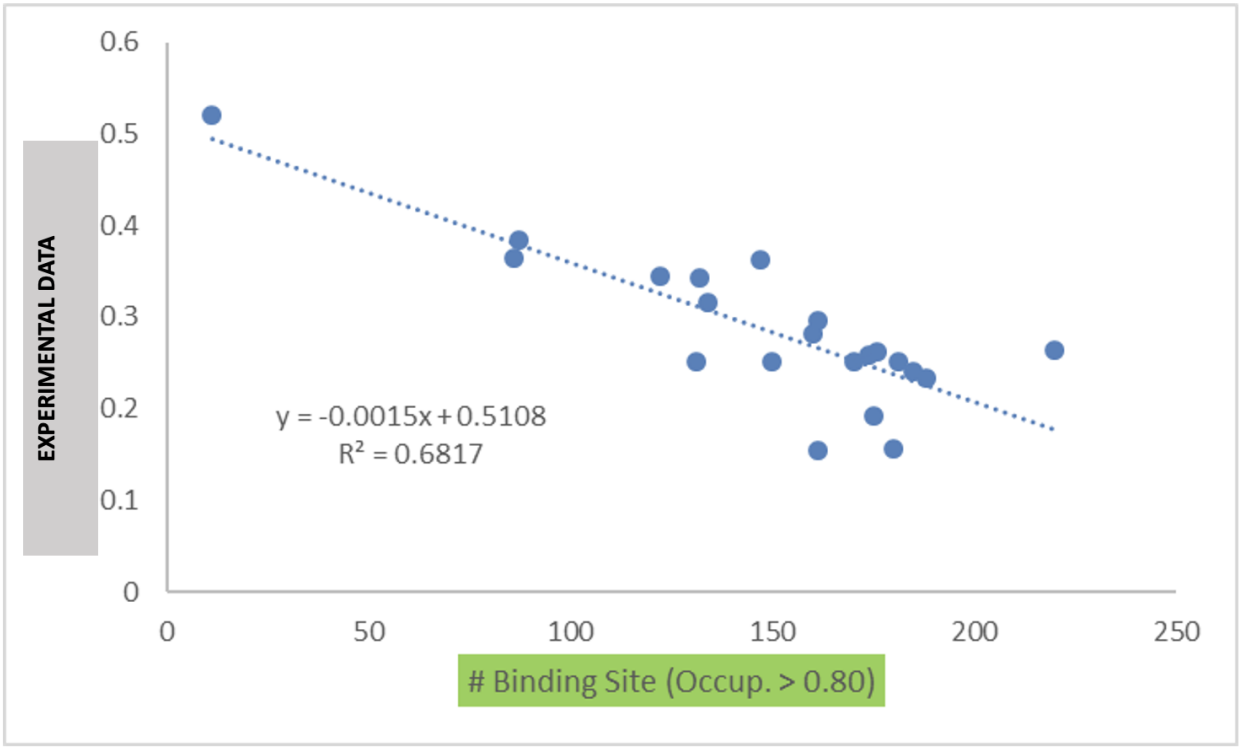
9.2.4 Data Interpretation

The data output from Step 4 can help connect experimental observables to the molecular details of protein-excipient interactions. In doing so, these simulation data can help both rationalize known trends for existing data on excipients and suggest the choice of new excipients for additional testing. With regard to the former, the most straightforward approach is to compare available experimental data to the Step 4 output and note correlations:

Excipient	# Binding Site (LGFE < -1)	# Binding Site (LE < -0.25)	# Binding Site (PPIP > 0.10)	# Binding Site (LE < -0.20 and PPIP > 0.10)	Experimental Data
trehalose	86	61	12	11	
alanine	210	207	38	37	
isoleucine	197	193	32	31	
arginine	129	90	23	14	
methionine	161	159	22	22	
glutamate	113	108	15	14	
proline	203	176	36	32	
sucrose	87	62	15	9	
lysine	135	131	22	21	
Excipient					

Please see [27] for a real-world example.

Below is an example of blinded data from a collaboration case study that demonstrates the utility of the binding site occupancy metrics computed by SILCS-Biologics and available in `report_output.xlsx`:



9.2.5 Visualizing SILCS FragMaps and SILCS-PPI Results

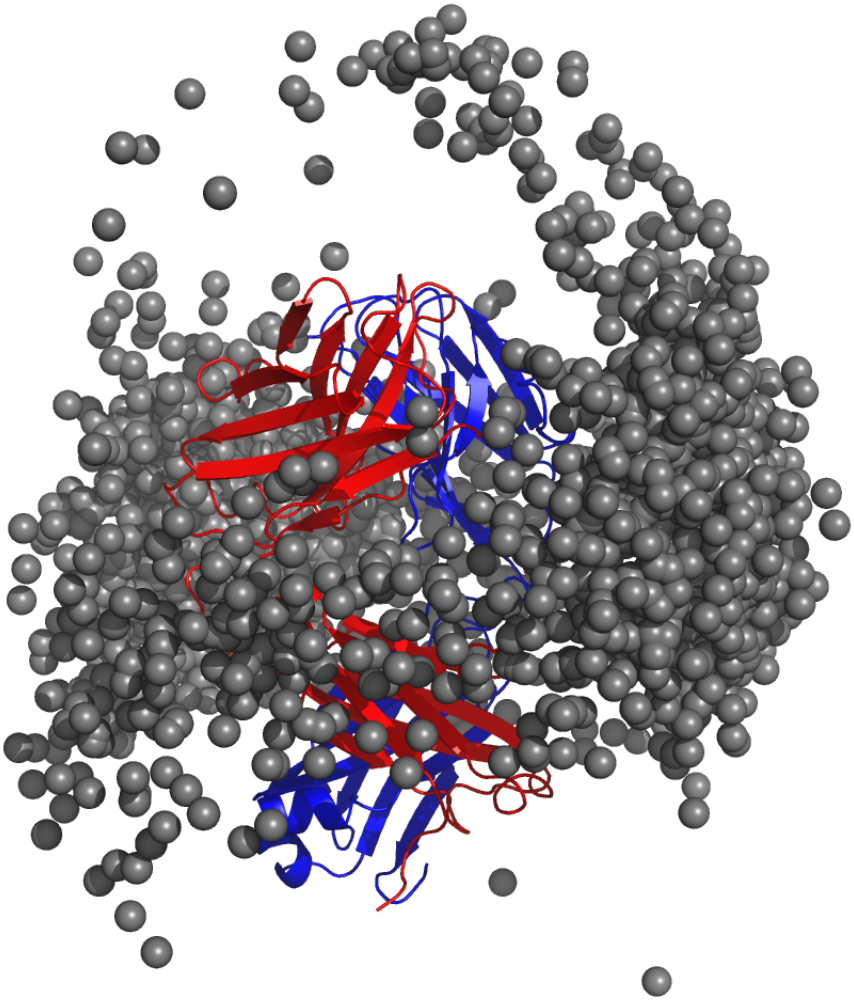
Tip: If you ran your SILCS-Biologics workflow using the SilcsBio GUI, SILCS FragMaps and SILCS-PPI results visualization will be available to you directly in the SilcsBio GUI upon successful completion of your SILCS-Biologics job. Please refer to *SILCS-Biologics Using the GUI* for more information.

SILCS FragMaps generated during Step 1 of the SILCS-Biologics workflow can be easily visualized using your choice of MOE, PyMol, or VMD molecular graphics software packages. FragMaps will be located in `$WORKDIR/1_fragmap.$job_id/fab/silcs_fragmaps_fab`. Please see *Visualizing SILCS FragMaps* for detailed instructions.

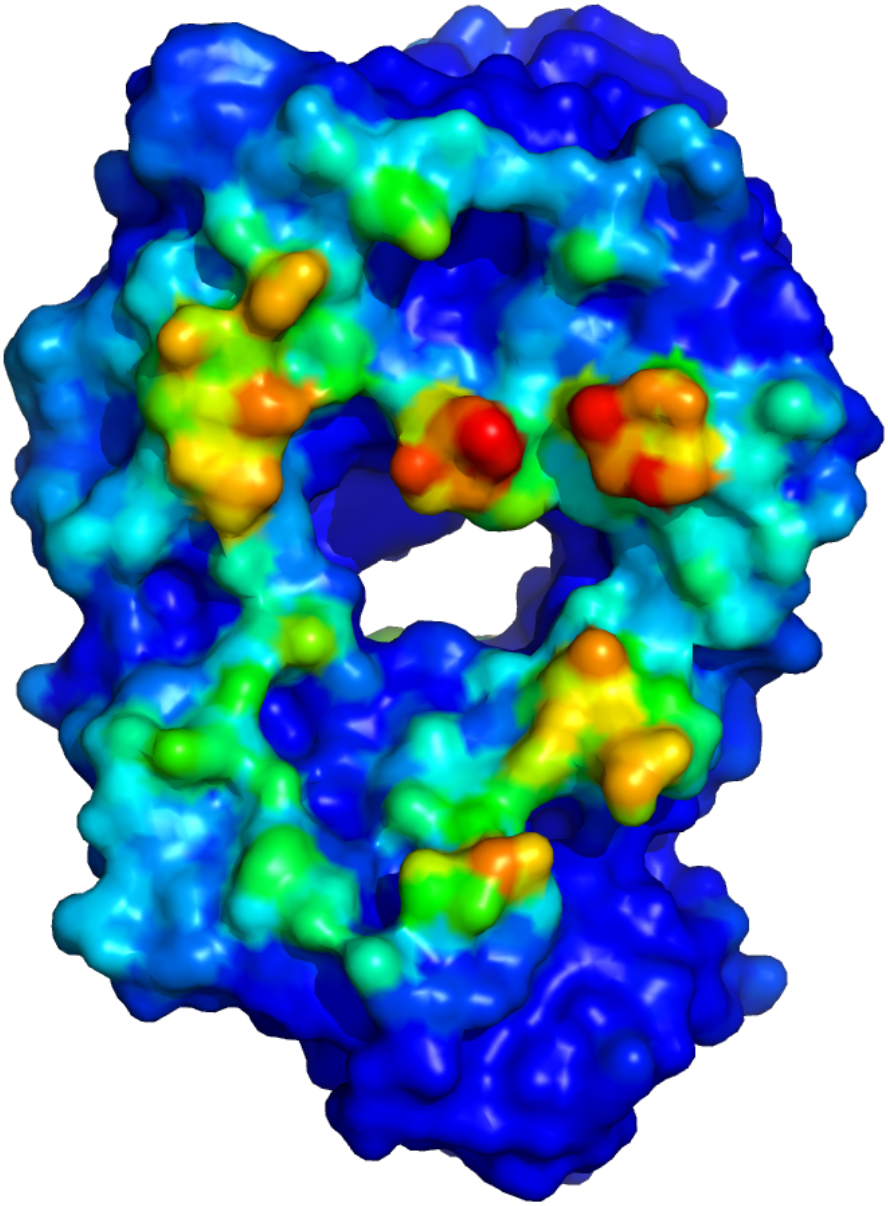
SILCS-Biologics Step 2 SILCS-PPI results can also be viewed using molecular graphics software. For the purposes of SILCS-PPI, one protein is considered the "receptor" and the other the "ligand". In the current use case with only one input protein domain, `prot1=fab.pdb`, `fab.pdb` data are used for both the "receptor" and the "ligand". For the purpose of SILCS-PPI, the ligand protein is docked to the receptor protein by comprehensively sampling locations and orientations of the ligand protein relative to the receptor protein. The relative locations and orientations are scored based on the overlap of the ligand functional groups and the receptor protein SILCS FragMaps (computed in Step 1). The docked poses of the ligand are then clustered as part of SILCS-PPI.

The `$WORKDIR/2_ppi.$job_id/fab_fab/3_ppi/receptor_clusters.pdb` file contains the

receptor protein structure and the cluster centroids. The cluster centroids in this file all have the same chain name, Z. Below is a molecular graphics image of `receptor_clusters.pdb`, with the receptor protein shown as ribbons and the cluster centroids as spheres. The CDR loops of the Fab are pointing to the top of the image, heavy chain amino acids are in red, light chain amino acids are in blue, and cluster centroids are in gray.



`$WORKDIR/2_ppi.$job_id/fab_fab/3_ppi/receptor_surf_contact.pdb` contains PPI propensity values mapped onto the receptor protein and recorded in the B-factor column, which provides an easy way to visualize which residues are most at risk for contributing to PPI, and, therefore, to aggregation of the biologic in its native (folded) state:



9.3 SILCS-Biologics CLI Workflows

In what follows, we discuss three use cases ordered by increasing complexity. The first case involves a protein small enough to be run intact. We use as an example a single Fab fragment (~450 amino acids) from an antibody. The second use case involves a hypothetical fusion protein engineered by combining the sequences of two separate 500 amino acid proteins, with each one forming one domain of the fusion protein. In this second example, the full length protein is first split into its two domains, and each domain is processed as a separate input for computational expediency. The third example is a complete antibody molecule (~1300 amino acids). It is split into three domains: the two Fab regions and the one Fc region.

The three use cases are ordered by increasing complexity. In the first case, only Fab-Fab PPI needs to be considered. Contrast this to the third, where FabA-FabA, FabA-FabB, FabA-Fc, FabB-FabB, FabB-Fc, and Fc-Fc PPI need to be considered. Despite this increased complexity, and the resulting need to keep track of multiple different simulations in the second and third use cases, SILCS-Biologics is easy to use in all three cases because it automatically manages all of the necessary simulations and resulting data.

9.3.1 Running the Complete Workflow with a Single Command

SILCS-Biologics is designed to be self contained, allowing all calculations across all four steps to be performed with a single command. SILCS-Biologics can also be used in a modular manner, which allows the user to run each step to completion and inspect intermediate results before moving on to the next step. The three use case examples below demonstrate its application in a self-contained manner. Please make sure you read through and understand these uses case examples. Details of its modular use follow after these use case examples.

Use Case 1. One Input Protein Domain

The simplest example with one input protein domain requires two input parameters, `step=1*2*3*4*` and `prot1=fab.pdb`. The first parameter requests that all four steps, including any substeps as indicated by the use of `*`, be run. The second parameter says to use use `fab.pdb` as the input protein domain file. `fab.pdb` contains coordinates for only the Fab portion of an antibody. To run this example, you can copy `fab.pdb` from `$SILCSBIODIR/examples/biologics/nist_fab/` to your local directory and run the following command:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1*2*3*4* \
    prot1=fab.pdb
```

Tip: Running the complete SILCS-Biologics workflow is a compute-intensive task. You can expect the `fab.pdb` example here to take 2 to 4 days total on a compute cluster with 10 GPU-enabled

nodes. Step 1 (SILCS) will take 1 to 2 days and Step 2 (SILCS-PPI) and Step 3 (SILCS-Hotspots) will take 0.5 to 1 day each.

Tip: Step 2 (SILCS-PPI) is a RAM-intensive task. For the `fab.pdb` example here, a single SILCS-PPI job will require about 5 GB of RAM, and will fail if not enough RAM is available. You may need to adjust the job control parameters in `$SILCSBIODIR/templates/ppi/run.tmpl` to ensure that your PPI jobs will have enough RAM to successfully run.

Tip: If you are unable to leave your terminal window open for the full duration of the `silcs-biologics` workflow, you can reply `y` when `silcs-biologics` asks "Do you want to run the workflow in the background using nohup?". This will launch `silcs-biologics` as a background job and allow it to keep running even if you logout from or close your terminal window. When you log back in, you can check the files `job_progress.$job_id` and `silcs-biologics_main.$job_id.log` (see below for details on how `$job_id` is set).

By default, `silcs-biologics` uses the excipient molecules in `$SILCSBIODIR/data/excipients/mols/`: alanine, arginine, aspartate, citrate, glucose, glutamate, glycine, histidine, lactate, lysine, malate, mannitol, phosphate, proline, sorbitol, succinate, sucrose, threonine, trehalose, and valine. Each molecule is in Mol2 format. If you prefer to provide your own excipients, create a directory and place a Mol2 format file for each excipient you would like into that directory. Your Mol2 files must contain optimized three-dimensional geometries as well as correct atom types. Additional excipients and buffers can be found in the `amino_acid/`, `buffers/`, and `sugars/` subdirectories within `$SILCSBIODIR/data/excipients/`. For example, you could create a directory `my_excipients/` in your working directory where you will run the `silcs-biologics` command, copy mol2 files of your choice from `$SILCSBIODIR/data/excipients/` into `my_excipients/`, and provide the optional parameter `molsdir=my_excipients` to run using these excipients:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1*2*3*4* \
    prot1=fab.pdb \
    molsdir=my_excipients
```

It is possible to differentiate excipients from the buffer. Without this distinction, all of the provided Mol2 files are posed and scored using SILCS-Hotspots, and the final report includes Ligand Grid Free Energies (LGFEs) for each Mol2. If a buffer Mol2 file is specified, it is likewise posed and scored using SILCS-Hotspots. However, the final reporting is done relative to the buffer. That is, the buffer molecule is not included in the reporting and each score for the non-buffer molecules is computed relative to the buffer molecule. For example, if you wish to use `phosphate.mol2` as the buffer molecule, you can include it in your `my_excipients/` directory and indicate it with the option `buffer=my_excipients/phosphate.mol2`:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics
    step=1*2*3*4* \
    prot1=fab.pdb \
    molsdir=my_excipients \
    buffer=my_excipients/phosphate.mol2
```

Tip: When using the SilcsBio GUI to run SILCS-Biologics, designation of an excipient molecule as buffer will be available once the compute-intensive parts of the workflow have successfully finished and the results are ready for analysis. Do make sure that any molecule you intend to analyze as a buffer is included with your other excipient molecules when you set up and perform your SILCS-Biologics workflow using the the GUI.

Use Case 2. Two Input Protein Domains

In this example, we start with a full-length structure of a hypothetical protein, `fullpdb.pdb`, with two independent domains, domain **X** and **Y**. To begin, you must create two input protein domain files corresponding to each domain.

To do so, simply make two copies of `fullpdb.pdb` and name one `protx.pdb` and the other `proty.pdb`. Then, edit `protx.pdb` and delete the amino acids corresponding to the domain Y. Repeat the same process for `proty.pdb`; edit `proty.pdb` and delete the amino acids corresponding to the domain X. Make sure that there is no overlap between the amino acids contained in `protx.pdb` and `proty.pdb`. In general, all amino acids in `fullpdb.pdb` should be accounted for by the combination of `protx.pdb` and `proty.pdb`; however, if a long flexible peptide connects `protx.pdb` to `proty.pdb`, the amino acids in that peptide region can be excluded from `protx.pdb` and `proty.pdb` for computational expediency with likely minimal impact on the final results.

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1*2*3*4* \
    prot1=protx.pdb \
    prot2=proty.pdb \
    fullpdb=fullpdb.pdb
```

Note that there are now two additional input parameters relative to Use Case 1: `prot2=proty.pdb` and `fullpdb=fullpdb.pdb`. The addition of `prot2=` indicates a second input protein domain must be considered and the addition of `fullpdb=` provides a reference structure for collating PPI contact data as well as for excluding surface-exposed amino acids in `protx.pdb` and `proty.pdb` that are in fact buried in the context of `fullpdb.pdb`. This latter point is important for both the SILCS-PPI and SILCS-Hotspots analysis to ensure that buried amino acids in `full.pdb` are not incorrectly noted as either contributing to PPI or having hotspots. Both of the additional input parameters, `prot2=` and `fullpdb=`, are required.

Note: If you do not have the full-length structure, but only have the structures of individual domains, then you will have to create the full-length structure using external molecular modeling tools, such as homology modeling software or simple alignment to a known full-length homologous crystal structure, to utilize this workflow. Save the resulting full-length structure as `fullpdb.pdb` and create `protx.pdb` and `proty.pdb` as described at the beginning of this example.

As with Use Case 1, you may specify a directory containing a custom set of excipients by adding `molsdir=<path to my excipient directory>` and/or rank the excipients relative to a buffer molecule by adding `buffer=<path to my buffer mol2 file>`.

Use Case 3. Three Input Protein Domains

A complete antibody molecule is a large protein, consisting of ~1300 amino acids. Additionally, for the purposes of molecular dynamics simulations, it requires a very large simulation box for solvation because of its extended Y-shaped conformation. Splitting it into three domains, specifically its two Fab regions and the one Fc region, makes the molecular dynamics-based SILCS simulations substantially more computationally tractable. Not only are the individual domains each ~1/3 the size of the full antibody, but also, when considered individually, the Fab and Fc regions are very compact and therefore can be simulated inside relatively small simulation boxes to achieve appropriate solvation.

We start with a full-length structure of the antibody, `antibody.pdb`. From this file, you must create three input protein domain files corresponding to the two Fab regions and the one Fc region, which we will call `faba.pdb`, `fabb.pdb`, and `fc.pdb`, respectively. Make three copies of `antibody.pdb` and name one `faba.pdb`, another `fabb.pdb`, and the third `fc.pdb`. Then, edit `faba.pdb` and delete the amino acids corresponding to the second Fab and the Fc regions. Edit `fabb.pdb` and delete the amino acids corresponding to the first Fab and the Fc regions. And edit `fc.pdb` and delete the amino acids corresponding to the first Fab and the second Fab regions. Make sure that there is no overlap between the amino acids contained in `faba.pdb`, `fabb.pdb` and `fc.pdb`.

In general, all amino acids in `fullpdb.pdb` should be accounted for by the combination of `faba.pdb`, `fabb.pdb`, and `fc.pdb`; however, if a long flexible a peptide connects `faba.pdb`, `fabb.pdb`, and/or `fc.pdb`, the amino acids in that peptide region can be excluded from `faba.pdb`, `fabb.pdb`, and `fc.pdb` for computational expediency with likely minimal impact on the final results. An example can be found in `$SILCSBIODIR/examples/nist_mab/` folder.

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1*2*3*4* \
    prot1=faba.pdb \
    prot2=fabb.pdb \
    prot3=fc.pdb \
    fullpdb=antibody.pdb
```

Note that there is one additional required input parameter relative to Use Case 2: `prot3=fc.pdb`.

The addition of `prot3=` indicates a third input protein domain will be considered. As with Use Case 2, `fullpdb=` indicates a reference structure for collating PPI contact data as well as for excluding surface-exposed amino acids in individual input protein domains that are in fact buried in the context of `antibody.pdb`. All of the input parameters in the above example are required.

Note: If you do not have the full-length antibody structure, but only have the structures of Fab and Fc domains, then you will have to create the full-length structure using other molecular modeling tools, such as homology modeling software.

Alternatively, you can align the domains onto other full-length IgG structures (e.g., PDB:1HZH from RCSB database). Save the resulting full-length structure as `fullpdb.pdb` and create `faba.pdb`, `fabb.pdb`, and `fc.pdb` as described at the beginning of this example.

As with the other use cases, you may specify a directory containing a custom set of excipients by adding `molsdir=<path to my excipient directory>` and/or rank the excipients relative to a buffer molecule by adding `buffer=<path to my buffer mol2 file>`.

9.3.2 Running the Workflow One Step at a Time

`silcs-biologics` can be used to run the SILCS-Biologics workflow in a stepwise fashion, with `step=1*` requesting only the SILCS simulations be run, `step=2*` requesting SILCS-PPI be run, `step=3*` requesting SILCS-Hotspots be run, and `step=4*` requesting processing of data from the prior steps and generation of the final report. Finer control at the level of the smaller substeps can also be requested, as detailed in the following example.

Stepwise Use Case 1. One Input Protein Domain

In this example, we use `fab.pdb` as the input protein domain. You can find this file in `$SILCSBIODIR/examples/biologics/nist_fab/`.

1. Step 1: Run SILCS and generate FragMaps

   You can run all the substeps of Step 1 automatically:

   ```
   $SILCSBIODIR/silcs-biologics/silcs-biologics \
       step=1* \
       prot1=fab.pdb \
       job_id=try01
   ```

   The `job_id=` parameter is used to group together job inputs and outputs from different steps/substeps. Therefore, when using `silcs-biologics` in a stepwise fashion, you will need to provide the same value for this parameter across all steps/substeps for the system you are modeling.

Alternatively, you can run substep by substep:

- Step 1a: Set up SILCS simulations

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1a \
    prot1=fab.pdb \
    job_id=try01
```

- Step 1b: Run SILCS simulations

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1b \
    prot1=fab.pdb \
    job_id=try01
```

  These SILCS simulations for `fab.pdb` will take 1 to 2 days on a cluster with 10 GPU-enabled compute nodes. If the simulation jobs fail due to external factors such as a power outage, server maintenance, etc., you can use the exact same command to resume the SILCS jobs from the point where they failed (as opposed to needing to restart them from the very beginning).

- Step 1c: Generate SILCS FragMaps

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1c \
    prot1=fab.pdb \
    job_id=try01
```

2. Step 2: Run SILCS-PPI

   To continue with Step 2 using your outputs from Step 1, run your commands in the same directory where you ran your Step 1 commands and use the same `$job_id` you used for Step 1.

   You can run all the substeps of Step 2 automatically:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=2* \
    prot1=fab.pdb \
    job_id=try01
```

   Alternatively, you can run each substep by using the following commands:

   - Sub-step 2a: Run SILCS-PPI jobs

```
$SILCSBIODIR/silcs-biologics/silcs-biologics step=2a \
    prot1=fab.pdb \
    job_id=try01
```

- Sub-step 2b: Collect PPI results

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=2b \
    prot1=fab.pdb \
    job_id=try01
```

3. Step 3: Run SILCS-Hotspots

   To continue with Step 3 using your outputs from Step 2, run your commands in the same directory where you ran your Step 2 commands and use the same `$job_id` you used for Step 1 and Step 2.

   As described previously in *Running the Complete Workflow with a Single Command*, you can specify a custom set of excipient molecules using the optional `molsdir=` parameter.

   You can run all the substeps of Step 3 automatically:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=3* \
    prot1=fab.pdb \
    job_id=try01
```

   Alternatively, you can run each substep by using the following commands:

   - Step 3a: Run excipient docking

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=3a \
    prot1=fab.pdb \
    job_id=try01
```

   - Step 3b: Cluster the hotspots

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=3b \
    prot1=fab.pdb \
    job_id=try01
```

4. Step 4: Collate and analyze data from prior steps and generate report

   The SILCS-Biologics data can be processed into a web report or a spreadsheet report. As described previously in *Running the Complete Workflow with a Single Command*, you can

specify a buffer molecule that will be used as a reference for ranking of the excipients using the optional `buffer=` parameter.

- Generate a web report

```
$SILCSBIODIR/silcs-biologics/silcs-biologics
    step=4a \
    prot1=fab.pdb \
    job_id=try01
```

- Run step 4b only (Spreadsheet_Report)

```
$SILCSBIODIR/silcs-biologics/silcs-biologics
    step=4b \
    prot1=fab.pdb \
    job_id=try01
```

Stepwise Use Case 2. Two Input Protein Domains

Follow the instructions for *Use Case 1. One Input Protein Domain*, and, in addition to `prot1=`, provide values for `prot2=` and `fullpdb=`.

Stepwise Use Case 3. Three Input Protein Domains

Follow the instructions for *Use Case 1. One Input Protein Domain*, and, in addition to `prot1=`, provide values for `prot2=`, `prot3=`, and `fullpdb=`.

9.3.3 Conserving Computing Resources

Re-Running a System with a Different Set of Excipients

If, after having run the SILCS-Biologics workflow, you decide you would like results for additional excipients, you can simply reuse your existing results from Step 1 (SILCS) and Step 2 (SILCS-PPI) without re-running these two steps. To do so, you will need to use a new `$job_id` for the new set of excipients. Let us assume your original simulations were in `$WORKDIR` and had the `$job_id` value `try01`. After initially completing the SILCS-Biologics workflow, you would have the following directories:

```
$WORKDIR/1_fragmap.try01
$WORKDIR/2_ppi.try01
$WORKDIR/3_excipients.try01
$WORKDIR/4_report.try01
```

To re-use your existing SILCS FragMap and SILCS-PPI data, copy the contents of their respective directories and associate the new directories with a new `$job_id`, `try02`:

```
cd $WORKDIR
cp -r 1_fragmap.try01 1_fragmap.try02
cp -r 2_ppi.try01 2_ppi.try02
```

Tip: To save disk space, you may create symbolic links to instead of making copies of your existing data.

```
cd $WORKDIR
ln -s 1_fragmap.try01 1_fragmap.try02
ln -s 2_ppi.try01 2_ppi.try02
```

However, be mindful that with symbolic links any changes you make to `1_fragmap.try02` or to `2_ppi.try02` (including deleting files) will also be made to `1_fragmap.try01` and `2_ppi.try01`. Therefore, we strongly recommend you use `cp -r` instead of `ln -s` if you have disk space available.

Now, re-run Step 3 and Step 4 using `job_id=try02`:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics
    step=3*4* \
    prot1=fab.pdb \
    molsdir=my_excipients_new
```

The above command will use excipient files contained in `$WORKDIR/my_excipients_new` for running the SILCS-Hotspots calculations and creating reports, and these results will be in the new directories `3_excipients.try02` and `4_report.try02`, respectively. If you like, you can also add the `buffer=` option. This same approach will also work for two or three input protein domains. Simply use the `prot2=`, `prot3=`, and `fullpdb=` options as you used for your initial `job_id=try01` run through the SILCS-Biologics workflow.

Conserving Computing Resources for Antibody Simulations

The most straightforward way to apply SILCS-Biologics to an antibody is to follow the directions for *Use Case 3. Three Input Protein Domains*, and we strongly recommend that new users use that approach. That said, it is possible to conserve computing resources by taking advantage of the fact that for a normal antibody (i.e., not a bi-specific antibody), the amino acid composition of the two Fab regions is identical. In other words, the amino acid sequence in `faba.pdb` is identical to `fabb.pdb`, and their structures are therefore also very similar. As such, a single set of SILCS FragMaps can be used for both Fab regions instead of computing FragMaps independently for both `faba.pdb` and for `fabb.pdb`.

Similar to *Use Case 3. Three Input Protein Domains*, you will have to create `faba.pdb`, `fabb.pdb`, and `fc.pdb` files from the full-length antibody structure, `fulllength.pdb` before we begin.

1. Generate FragMaps for one Fab domain:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1* \
    prot1=faba.pdb \
    job_id=try01
```

2. Generate FragMaps for the Fc domain (this can be done in parallel with the step 1):

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=1* \
    prot1=fc.pdb \
    job_id=try01
```

3. Generate FragMaps for the other Fab domain:

```
python $SILCSBIODIR/utils/python/reorient_maps.py faba.pdb fabb.pdb \
    1_fragmap.try01/faba/silcs_fragmaps_faba/maps \
    --outdir 1_fragmap.try01/fabb/silcs_fragmaps_fabb/maps
```

4. Continue running the remaining steps from 2 to 4:

```
$SILCSBIODIR/silcs-biologics/silcs-biologics \
    step=2*3*4* \
    prot1=faba.pdb \
    prot2=fabb.pdb \
    prot3=fc.pdb \
    fullpdb=antibody.pdb
```

SILCS-Biologics Directory Structure

`silcs-biologics` creates the below directories in `$WORKDIR`. By default, `$WORKDIR` is the directory in which `silcs-biologics` was run. Otherwise, `$WORKDIR` is determined by the parameter passed to `silcs-biologics` through the command line option `workdir=`.

Step 1 SILCS FragMap results will be in `1_fragmap.$job_id/`,

```
                                |>>> 1_setup
                    |>>> $prot1 >>> |>>> 2a_run_gcmd
                    |               |>>> 2b_gen_maps
                    |               |>>> silcs_fragmaps_$prot1
```

(continues on next page)

(continued from previous page)

```
1_fragmap.$job_id >>> |>>> $prot2 ...
                      |>>> $prot3 ...
```

Step 2 SILCS-PPI results will be in `1_ppi.$job_id/`,

```
                                          |>>> maps1
                  |>>> $prot1_$prot1 >>> |>>> maps2
                  |                       |>>> 3_ppi
2_ppi.$job_id >>> |>>> $prot1_$prot2 ...
                  |>>> $prot1_$prot3 ...
                  |>>> $prot2_$prot1 ...
                  |>>> $prot2_$prot2 ...
                  |>>> $prot2_$prot3 ...
                  |>>> $prot3_$prot1 ...
                  |>>> $prot3_$prot2 ...
                  |>>> $prot3_$prot3 ...
```

Step 3 SILCS-Hotspots excipient docking results will be in `3_excipients.$job_id/`,

```
                           |>>> mols
                           |>>> $prot1 >>> 4_hotspots >>> ***
3_excipients.$job_id >>>   |>>> $prot2 ...
                           |>>> $prot3 ...
```

and Step 4 reporting will be in `4_report.$job_id`,

```
4_report.$job_id
```

9.4 SILCS-PPI

Protein-Protein Interactions (PPI) energy function based on the Site-Identification by Ligand Competitive Saturation (SILCS) framework and utilize the fast Fourier transform (FFT) correlation approach. The free energy content of the SILCS FragMaps represent an alternative to traditional energy grids and they can be efficiently utilized to guide FFT-based protein docking.

9.4.1 SILCS-PPI Using the SilcsBio GUI

Coming soon!

9.4.2 SILCS-PPI Using the CLI

The SILCS-PPI workflow can be run with the following steps:

1. **Set up and run PPI calculations:**

   User must have completed the SILCS simulations and generated FragMaps for the target proteins. If self interactions are to be considered, the same protein must be specified for both `prot1` and `prot2`. Typically, the first protein is the "receptor" and the second protein is the "ligand".

```
$SILCSBIODIR/ppi/1_setup_ppi prot1=<first prot pdb> mapsdir1=
↪<location and name of directory containing FragMaps> prot2=<second␣
↪prot pdb> mapsdir2=<location and name of directory containing␣
↪FragMaps>
```

   Required parameters:

   - Path and name of first protein PDB file:

```
prot1=<first protein PDB>
```

   - Path and name of directory containing FragMaps for the first protein:

```
mapsdir1=<location and name of directory containing FragMaps>
```

   - Path and name of second protein PDB file:

```
prot2=<second protein PDB>
```

   - Path and name of directory containing FragMaps for the second protein:

```
mapsdir2=<location and name of directory containing FragMaps>
```

   Optional parameters:

   - Path and name of directory for PPI results:

```
ppidir=<folder where PPI result is stored; default: 3_ppi>
```

   - Rotation step size for sampling second protein orientation in reference to the first protein:

```
rotsize=<rotation step size; 10/15/20; default=10>
```

A pre-defined set of relative transformations based on the `rotsize` parameter are selected from the `$SILCSBIODIR/data/ppi_rot_angles` directory.

- Number of solutions per rotation:

```
numsol=<number of solutions per rotation; default=10>
```

- Bundle multiple (single) jobs into larger jobs:

```
bundle=<bundle multiple (single) jobs into larger jobs;␣
↪true/false; default=true>
```

If `bundle=true`, the number of jobs is determined by the `njobs` parameter. If `bundle=false`, each job will be submitted separately.

- Total number of jobs used when `bundle=true`:

```
njobs=<total # of jobs used when bundle=true; default=16>
```

- Number of processors per job:

```
nproc=<number of processors per job; default=8>
```

- Restart a job for specific bundle:

```
restart=<restart a job for specific bundle; e.g. restart=
↪#1; default=false>
```

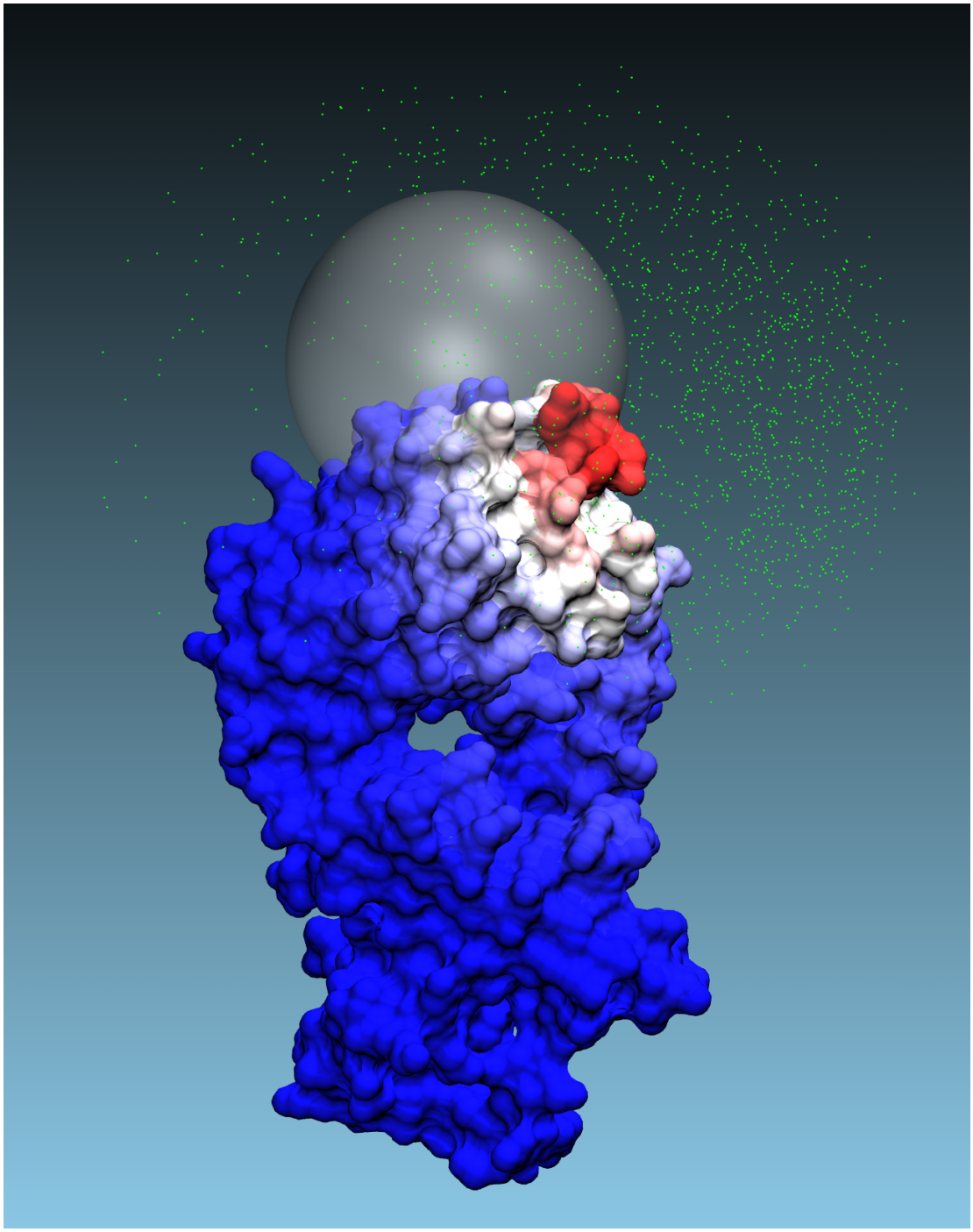
If `restart=#<bundle_number>`, the job will be restarted for the specified bundle number.

Additional parameters:

- Using Full Size of fragmap for PPI calculations:

```
fullmap=<flag for using full map or not; true/false;␣
↪default=true>
```

If `fullmap=false`, the center and radius parameters must be set.



- Center for transformation parameter:

```
center="x,y,z"; e.g. center="20,30,-2"
```

The center is the point on the surface of the first protein around which a sphere of radius `radius` will be defined. Only the poses of the second protein that are interfacing with at least one of residues of the first protein within the specified radius will be considered for PPI calculations. If `fullmap=false`, this parameter must be set.

- Radius for transformation parameter:

```
radius=<X Angstroms>
```

The solutions will be restricted to the points on the surface of the first protein that are within the specified radius from the center. If `fullmap=false`, this parameter must be set.

- Truncate maps and remove overlaps:

```
filter=<truncate maps and remove overlaps; true/false;␣
↪default=true>
```

If `filter=true`, the maps will be truncated by setting the GFE values to zero for all points that are not within the cutoff distance from the surface of the exclusion map. If `filter=false`, the maps will not be truncated and all points will be used in the PPI calculations which may lead to longer runtimes and larger output files.

- Cutoff for filtering:

```
cutoff=<cutoff for filtering; default=5>
```

The cutoff distance is used to determine which points are within the range from the surface of the exclusion map. If `filter=true`, the GFE values for all points that are not within the cutoff distance from the surface of the exclusion map will be set to zero.

- Custom parameter file:

```
paramsfile=<custom parameter file; default: $SILCSBIODIR/
↪templates/ppi/params.tmpl>
```

The custom parameter file can be used to override the default parameters used for PPI calculations. If not specified, the default parameter file will be used.

- Path to Python executable:

```
python=<path to python executable for filtering maps; default=
↪$(which python)>
```

The path to the Python executable used for filtering maps. If not specified, the default Python executable identified by using the `which python` command will be used.

2. **Collect PPI results and perfor clustering:**

Once the PPI calculations are complete, the results can be collected and clustered to identify the most favorable binding poses of the second protein relative to the first protein.

```
$SILCSBIODIR/ppi/2_collect_ppi prot1=<first prot pdb> prot2=<second␣
↪prot pdb>
```

Required parameters:

- Path and name of first protein PDB file:

```
prot1=<first protein PDB>
```

- Path and name of second protein PDB file:

```
prot2=<second protein PDB>
```

Optional parameters:

- Path and name of directory where PPI results are stored:

```
ppidir=<folder where PPI result is stored; default: 3_ppi>
```

- Perform clustering:

```
cluster=<perform clustering; true/false; default=true>
```

If `cluster=true`, the PPI results will be clustered to identify the most favorable binding poses of the second protein relative to the first protein.

- Number of top clusters to output:

```
topn=<number of top clusters to output; default=2000>
```

- Path to Python executable for clustering:

```
python=<path to python executable for clustering; default=
↪$(which python)>
```

3. **Re-build PPI complex PDB files:**

After the PPI results have been collected and clustered, the final step is to rebuild the PPI complex PDB files for visualization and further analysis.

```
$SILCSBIODIR/ppi/3_build_cmpx_pdb prot1=<first prot pdb> prot2=
↪<second prot pdb>
```

Required parameters:

- Path and name of first protein PDB file:

```
prot1=<first protein PDB>
```

- Path and name of second protein PDB file:

```
prot2=<second protein PDB>
```

Optional parameters:

- Path and name of logfile from the previous step (2_collect_ppi):

```
logfile=<ppi logfile from 2_collect_ppi; default=ppips_clusters.
↪log>
```

- Prefix for output PDB files:

```
prefix=<prefix for output pdb files; default=complex>
```

CHAPTER

TEN

SSFEP SUITE

10.1 SSFEP: Single Step Free Energy Perturbation

10.1.1 Background

Free energy perturbation (FEP) has long been considered the gold standard in calculating relative ligand-binding free energies. However, FEP is often impractical for evaluating a large number of changes to a parent ligand due to the large computational cost. Single Step Free Energy Perturbation (SSFEP) is an alternative that can be orders of magnitude faster than conventional FEP when evaluating a large number of changes to a parent ligand, while maintaining useful accuracy for small functional group modifications [28].

The SSFEP method involves post-processing of MD simulation data of a ligand in a given environment in the canonical ensemble to estimate the alchemical free energy change of chemically modifying the ligand. The following FEP formula is used,

$$\Delta G^{\mathrm{env}}_{\mathrm{L1}\to\mathrm{L2}} = -k_{\mathrm{B}}T \ln \left\langle e^{-\beta \Delta E} \right\rangle_{\mathrm{L1}} \tag{10.1}$$

where k_{B} is the Boltzmann constant and T is the temperature. The angular brackets indicate an average of the exponential factor over the MD trajectory of ligand $L1$ in the given environment, `env`, which can be either the solvated protein or water. ΔE is the energy difference between the two systems involving L1 and L2, which in practice is computed as the difference in the interaction energies of the two ligands in the corresponding environment:

$$\Delta E = E^{\mathrm{env}}_{L2} - E^{\mathrm{env}}_{L1}$$

The environment `env` in each system is defined as all non-ligand atoms. Once $\Delta G^{\mathrm{protein}}_{L1\to L2}$ and $\Delta G^{\mathrm{water}}_{L1\to L2}$ are computed according to Eq. (10.1), the relative binding free energy is given by

$$\Delta\Delta G^{\mathrm{bind}}_{L1\to L2} = \Delta G^{\mathrm{protein}}_{L1\to L2} - \Delta G^{\mathrm{water}}_{L1\to L2}$$

The SSFEP approach allows the data from simulation of a single protein–ligand complex to be rapidly post-processed to evaluate tens to hundreds of potential modifications involving multiple

sites on the parent ligand. Given this, the best results are achieved when SSFEP is used to evaluate small modifications to the parent ligand.

The ability of standard FEP and SSFEP to reproduce the experimental relative binding affinities of known ligands for two proteins, ACK1 and p38 MAP kinase, was tested [29]; SSFEP was able to produce comparable results to full FEP while requiring a small fraction of the computational resources [29].

10.2 SSFEP Using the SilcsBio GUI

Please see *SSFEP Simulations Using the GUI* in the *Graphical User Interface (GUI) Quickstart* for instructions on running SSFEP from the SilcsBio GUI.

10.3 SSFEP Using the CLI

The following usage details are provided for completeness. **We strongly recommend using the SilcsBio GUI to set up, run, and analyze SSFEP calculations**.

To perform the SSFEP precomputation simulations, protein coordinates in PDB file format and parent ligand coordinates in Mol2 file format are required. The protein should have termini properly capped, missing loops built or the ends of the missing loops capped, standard atom and residue names, and sequential atom and residue numbering. Using these two files, run the following:

```
${SILCSBIODIR}/ssfep/1_setup_ssfep prot=<Protein PDB> lig=<Ligand Mol2/
↪SDF>
```

Warning: The setup program internally use the GROMACS utility `pdb2gmx`, which may have problems processing the protein PDB file. The most common `pdb2gmx` issue involves mismatches between the expected residue name/atom names in the input PDB and those defined in the CHARMM force-field.

To fix this problem: Run the `pdb2gmx` command manually from within the `1_setup` directory for a detailed error message. Please contact support@silcsbio.com for additional assistance.

Following completion of the setup, run 10 MD jobs:

```
${SILCSBIODIR}/ssfep/2_run_md_ssfep prot=<Protein PDB> lig=<Ligand Mol2/
↪SDF>
```

This command will submit 10 jobs to the pre-defined queue: 5 for the ligand in water and 5 for the ligand complexed with protein.

Once the precomputation simulations are completed, the `2_run_md/1_lig/[1-5]` and `2_run_md/2_prot_lig/[1-5]` directories will contain `*.1-10.whole.trr` trajectory files. If these files are not generated, then your simulations are either still running or have stopped due to a problem. Look into the log files within these directories for an explanation of the failure.

10.4 Ligand Modifications

Follow the instructions in *Chemical Group Transformations* to create modifications to your parent ligand.

10.5 Evaluating Binding Affinity Changes

Once `modifications.txt` has been prepared and the MD simulations involving the parent ligand are completed, run the following script to set up a $\Delta\Delta G$ calculation.

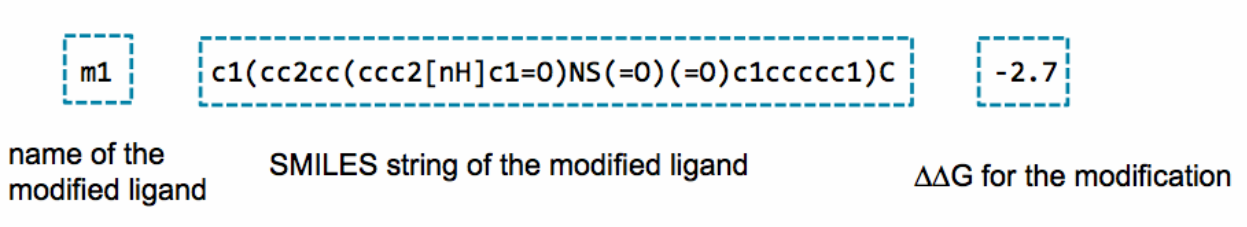
```
${SILCSBIODIR}/ssfep/3a_setup_modifications prot=<Protein PDB> lig=
↪<Ligand Mol2/SDF File> mod=modifications.txt
```

This will submit 10 jobs to evaluate all snapshots from the completed MD simulations of the parent ligand in order to calculate the change in free energy for every modification specified in your `modifications.txt`. Structures of these modifications in Mol2 format are output as `3_analysis_<modified ligand name entry in modifications.txt>/mod_files/*.mol2`.

After these jobs complete, you may obtain $\Delta\Delta G$ for your full list of modifications using:

```
${SILCSBIODIR}/ssfep/3b_calc_ddG_ssfep mod=modifications.txt
```

Example output follows:



CHAPTER

ELEVEN

CGENFF SUITE

11.1 CGenFF Suite Overview

The CGenFF Suite offers several utilities to enable molecular dynamics (MD) simulations of biomolecular systems. Force field topologies and parameters for novel compounds can be generated in seconds with the *CGenFF program*. These compounds can also be *covalently linked to amino acids* to model covalent protein–ligand interactions. The resulting topologies and parameters can be used as input for *standard MD simulations*, which can be used to extract dynamic structural and energetic data.

- **CGenFF Program**:

  Rapidly assign CHARMM General Force Field topologies and parameters for a wide range of drug-like molecules, enabling simulations of compounds in biological environments without extensive, time-consuming parameterization.

- **CGenFF for Covalent Ligands**:

  Generate CHARMM General Force Field topologies and parameters for amino acids covalently linked to ligands of interest.

- **Standard MD Simulations**:

  Set up, run, and analyze MD simulations of any protein of interest, including protein–ligand complexes and proteins in the presence of solutes.

11.2 CGenFF Program

11.2.1 Background

The CHARMM General Force Field (CGenFF) covers a wide range of chemical groups present in biomolecules and drug-like molecules, including a large number of heterocyclic scaffolds [30]. The parametrization philosophy behind the force field focuses on quality at the expense of transferability, with the implementation concentrating on an extensible force field.

The CGenFF program uses the CHARMM additive potential energy function to calculate the energy as a function of the Cartesian coordinates of the system, as shown below.

$$\text{BondedTerms}$$
$$\sum_{\text{bonds}} K_b(b-b_0)^2 + \sum_{\text{angles}} K_\theta(\theta-\theta_0)^2 + \sum_{\text{dihedrals}} K_\phi(1+\cos(n\phi-\delta))$$
$$+ \sum_{\text{improper}} K_\psi(\psi-\psi_0)^2 + \sum_{\text{Urey}-\text{Bradley}} K_{\text{UB}}(r_{1,3}-r_{1,3;0})^2$$

$$\text{NonbondedTerms}$$
$$\sum_{\text{nonbonded}} \frac{q_i q_j}{4\pi D r_{ij}} + \epsilon_{ij}\left[\left(\frac{R_{\text{min},ij}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{\text{min},ij}}{r_{ij}}\right)^{6}\right]$$

The bonded portion of the potential energy function consists of terms for the bonds, valence angles, torsion or dihedral angles, improper dihedral angles, and a Urey-Bradley term, where b_0, θ_0, ϕ_0, and $r_{1,3;0}$, respectively, are the bond, angle, improper and Urey-Bradley equilibrium values, the K's are the corresponding force constants, and n and δ are the dihedral multiplicity and phase. The nonbonded portion consists of an electrostatic term, with q_i and q_j being the respective partial atomic charges on atoms i and j, and a van der Waals (vdW) term, which is treated by the Lennard-Jones (LJ) 6-12 potential in which ϵ_{ij} is the well depth, $R_{\text{min},ij}$ is the radius, and r_{ij} is the distance between atoms i and j.

Simulation of typical molecular systems of interest requires large numbers of parameters. To make the assignment of these parameters practical, force fields require atom types to be assigned to all the atoms in the system, with the parameters associated with combinations of atom types. For instance, the parameter list will contain K_ϕ, n, and δ values for the dihedral parameters associated with all sequentially bonded combinations of four atom types that occur in the molecules supported by the force field. Thus, the first step of assigning parameters for a chemical system is assigning atom types to that system.

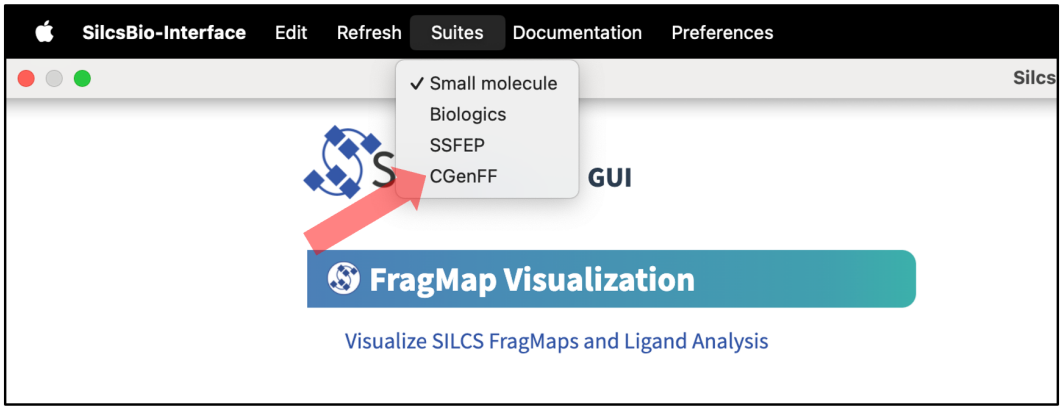
The CGenFF program employs an algorithm that can automatically assign atom types to a given molecule. The atom typer is rule-based and programmable, making it easy to implement complex atom typing rules and to update the atom typing scheme as the force field grows. In the event that the CGenFF force field file does not explicitly contain a bonded parameter required for simulating the molecule, the CGenFF program will assign bonded parameters by substituting analogous atom types in the definition of the desired parameter. A penalty is associated with every substitution and the existing parameter with the lowest total penalty is chosen as an approximation for the desired parameter. Charges are assigned using an extended bond-charge increment scheme that is able to capture inductive and mesomeric effects across up to 3 bonds. More details can be found in [31].

The CGenFF Suite provides the CGenFF program in both the SilcBio GUI and CLI formats. The SilcsBio GUI provides an intuitive option for users to parametrize their ligands. For advanced users, the CLI allows for more customization and additional features. The usage of CGenFF with the SilcsBio GUI and CLI are described below.

11.2.2 CGenFF Using the SilcsBio GUI

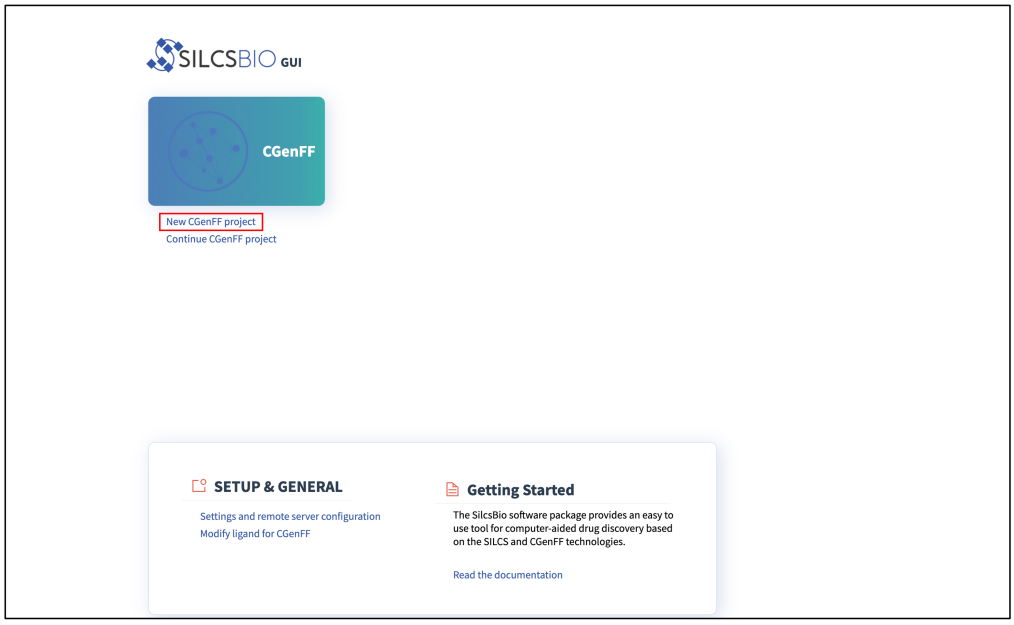
1. **Enter the CGenFF Suite:**

   As of Release v 2024.1, the SilcsBio GUI allows users to use CGenFF. The CGenFF Suite is acessed by selecting *Suites* → *CGenFF* from the menu bar.



2. **Begin a new CGenFF project:**

   Select *New CGenFF project* from the Home page.



3. **Enter a project name select the remote server:**

   Enter a project name and select the remote server where the CGenFF job will run. Input and output files from the CGenFF job will be stored on this server. For information on setting up the remote server, please refer to *Remote Server Setup*. Next, select a molecule Mol2 file. As described in *File and Directory Selection*, choose a file from your local computer ("localhost") or from any server you had configured through the *Remote Server Setup* process. For correct atom typing, it is important that all hydrogens are present, the system has correct protonation and tautomeric states, and that the bond orders are correct.

4. **Set up the CGenFF job:**

   Under "Advanced Settings", desired information can be specified in the CGenFF output through selecting the listed options.

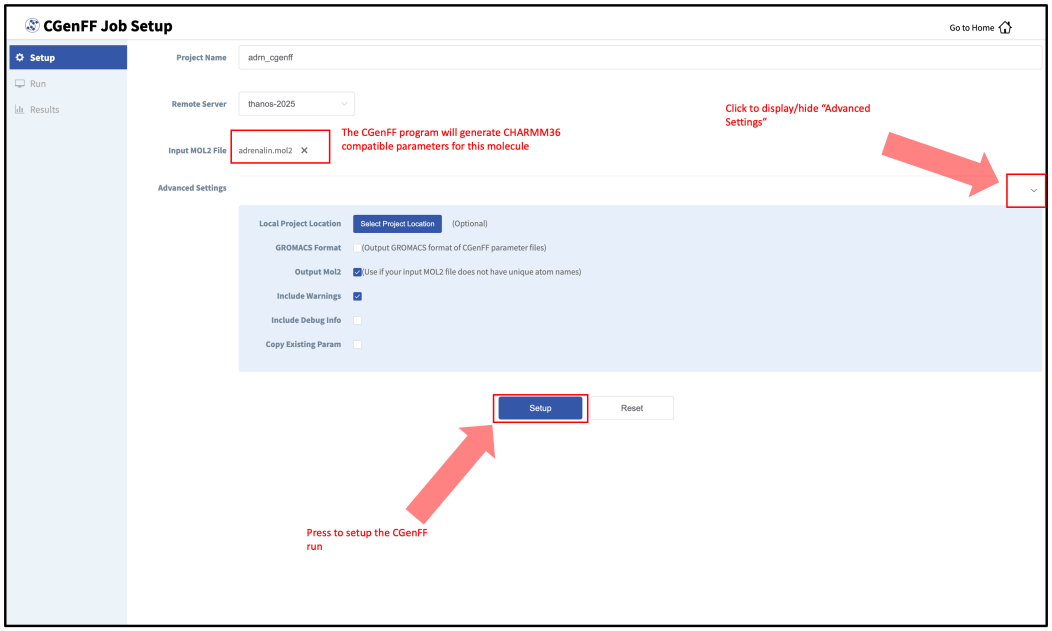
   The CGenFF assigned parameters, by default, are output in CHARMM compatible format. GROMACS compatible topology and parameter files can be specified by selecting the "GROMACS Format" option.

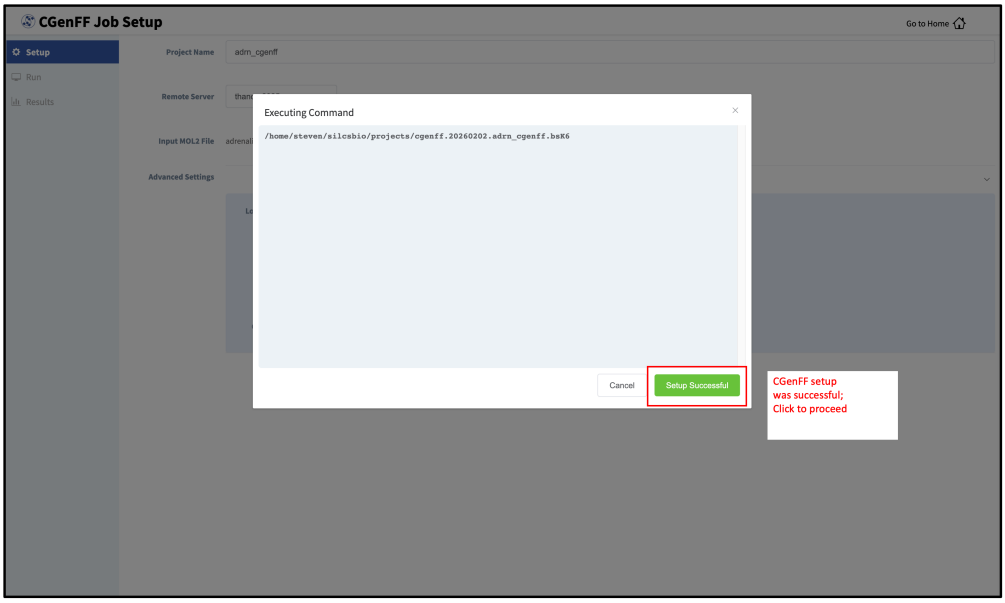
   If your input Mol2 does not have unique atom names, the CGenFF program will rename the atoms and the parameter file will reflect the names of the renamed atoms. In this case, it is advised to select the "Output Mol2" option.

   In the event that the input molecule results in an error, you may select the "Include Debug Info" to identify possible erroneous entries in the input Mol2 file (e.g., missing hydrogen atoms).

   By default, the CGenFF program will only output newly generated parameters as users are expected to use the force field in conjunction with the CHARMM36 force field. The parameters for which CGenFF directly applies to the input molecule can be explicitly listed in the output parameter file by selecting the "Copy Existing Param" option.

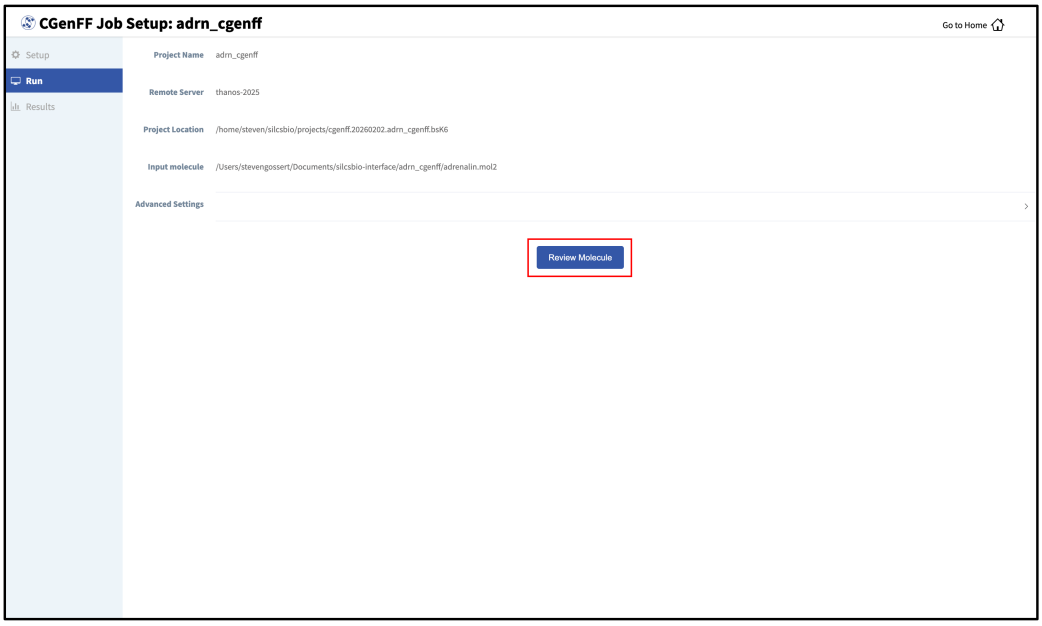
   After all information is specified, click the "Setup" button to proceed. Once the CGenFF job is set up, the GUI will indicate that the setup was successful by displaying a green "Setup Successful" button. Click the green "Setup Successful" button to continue.

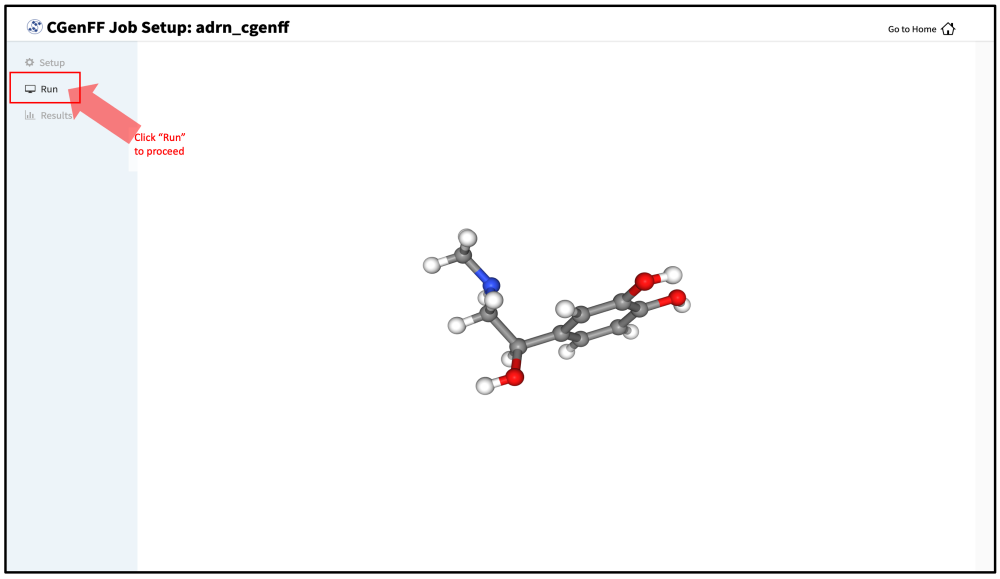




5. **Review the ligand:**

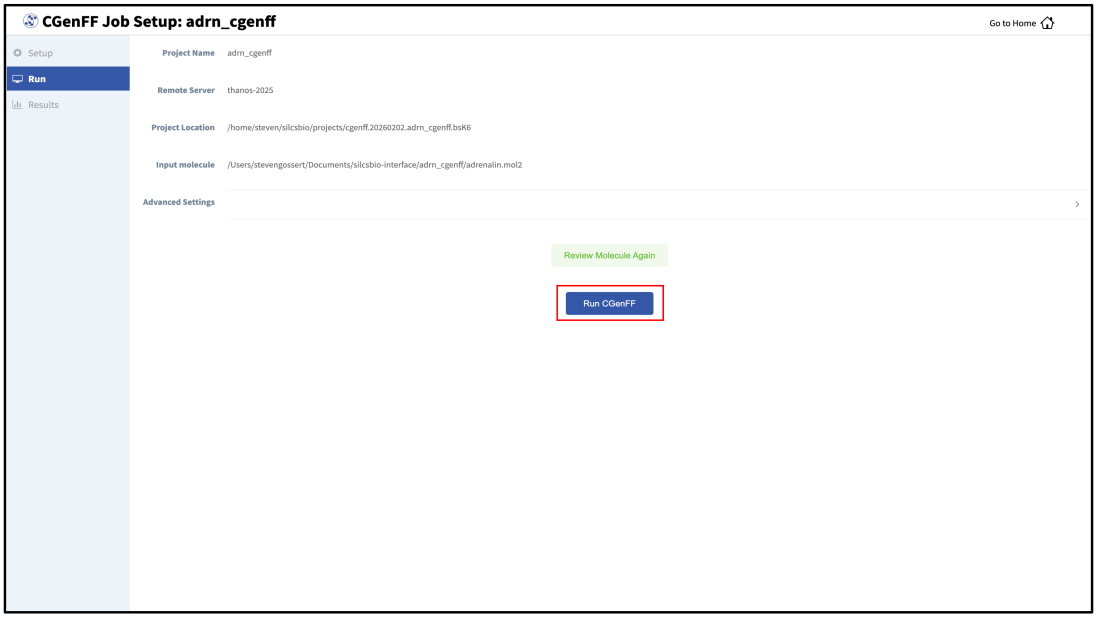
   After setting up the CGenFF run, the GUI will prompt you to review your molecule. Click the "Review Molecule" button at the bottom of the screen to visualize the molecule. After reviewing the molecule, click "Run" in the sidebar menu to continue.





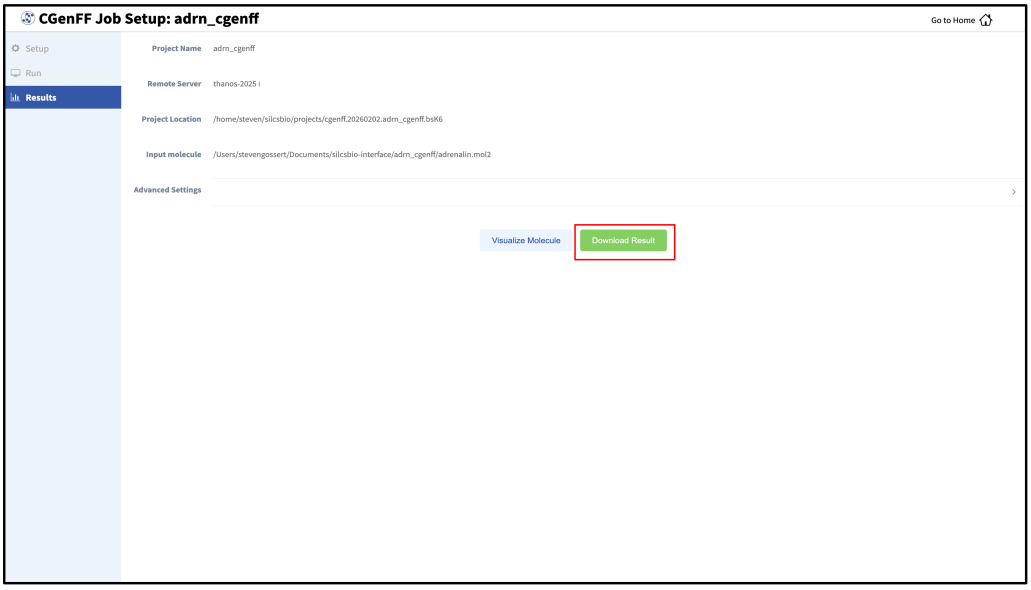
6. **Launch the CGenFF job:**

   To run the CGenFF program, click the "Run CGenFF" button at the bottom of the page. You may also review the molecule again using the "Review Molecule Again" button.



7. **Download CGenFF assigned topology and parameters:**

   Once the CGenFF program assigns force field parameters to the molecule, the results will be downloadable by clicking the green "Download Result" button.



For more information on the output stream file format and advanced CGenFF usage using CLI, please continue on to *CGenFF Using the CLI*. Please contact support@silcsbio.com if you need additional assistance.

11.2.3 CGenFF Using the CLI

The CGenFF program accepts Mol2 format files. For correct atom typing, it is important that all hydrogens are present, the system has correct protonation and tautomeric states, and that the bond orders are correct.

Once the Mol2 file is ready, the CGenFF program can be executed:

```
${SILCSBIODIR}/cgenff/cgenff <mol2file>
```

This produces a stream file in standard output. The output can be redirected to a `.str` file. The stream file consists of a topology file and the extra parameter file combined in a single output. It is up to the user to determine whether to use the stream file directly or to split the file into two separate files.

As an example, the adrenalin molecule is parametrized in CGenFF. The example input and output files can be found in `${SILCSBIO}/examples/cgenff/adrenalin.mol2` and `${SILCSBIO}/examples/cgenff/adrenalin.str`, respectively.

The output stream file has two major sections.

```
* Toppar stream file generated by
* CHARMM General Force Field (CGenFF) program version 3.0
* For use with CGenFF version 4.6
*

read rtf card append
... topology section ...

END

read param card flex append
... parameter section ...

END
RETURN
```

In CHARMM, asterisk and exclamation marks represents comments. If you wish to separate topology and parameter sections, you can take out the corresponding section after `read rtf` or `read param` to the `END` statement. Let's take a look at topology section:

```
RESI ADRN           0.000 ! param penalty= 113.000 ; charge penalty=  16.
↪607
GROUP             ! CHARGE   CH_PENALTY
ATOM O1     OG311  -0.649 !   11.757
ATOM O2     OG311  -0.530 !    0.000
```

(continues on next page)

(continued from previous page)

```
ATOM O3     OG311  -0.530 !    0.000
ATOM N      NG311  -0.524 !   16.607
ATOM C1     CG311   0.127 !   16.171
```

Above is the beginning of the topology section. This section defines the atom names and types, as well as the atomic charges. Each entry is followed by a charge penalty score. The atomic partial charges are assigned based on chemical analogy to existing parametrizations in the force field. The penalty score becomes high when a good analogy is not found.

Below is an excerpt from the parameter section. When an exact parameter is not found in an existing parameter database, an entry is added based on an analogous parameter. Similar to atomic partial charges, each entry in the parameter section is also followed by a penalty score.

```
ANGLES
CG2R61 CG311  OG311    75.70    110.10 ! ADRN , from CG2R61 CG321 OG311,␣
 ↪penalty= 4
CG311  CG321  NG311    43.70    112.20 ! ADRN , from CG331 CG321 NG311,␣
 ↪penalty= 1.5
CG321  NG311  CG331    40.50    112.20    5.00   2.42170 ! ADRN , from␣
 ↪CG3AM1 NG311 CG3AM1, penalty= 35
```

High penalty scores indicate that a parameter may be targeted for further optimization. Typically, penalty scores greater than 50 warrant further optimization.

It is possible to request that the CGenFF program preserve atomic partial charges from the input Mol2 file instead of assigning these values. This is done using the `-z` flag:

```
${SILCSBIODIR}/cgenff/cgenff -z <mol2file>
```

Warning: We STRONGLY advise against using your own custom atomic partial charges. Nonbonded interactions in the force field depend upon both the Coulomb and the Lennard-Jones terms, and the parameter values for these terms have been carefully developed together. Assigning custom atomic partial charges with the `-z` flag is likely to disrupt the balance in these nonbonded force field terms.

Additional options

- `-h` or `--help`: Displays help message and exits.
- `--version`: Displays version and licensing information, then exits.
- `-q` or `--quiet`: Suppresses all non-fatal warnings.
- `-v` or `--verbose`: Displays more warning messages. Use twice to also include debug information.
- `-f` or `--output-file` (filename): Default: - (stdout)
- `-m` or `--message-file` (filename): Determines where to write warning, error and debug messages. Default: - (errstr)
- `-a` or `--all-parameters`: Copies existing parameters to the output. This is of very limited use because the atom types and L-J are still missing.
- `-i` or `--input-format` (format): Determines format for input molecule. Currently, only the default (Mol2) format is supported, so this flag is of no use.
- `-w` or `--output-mol`: Outputs processed Mol2 file of the molecule after assigning parameters.
- `-x` or `--split-output` (prefix): splits the output into separate *.rtf* and *.prm* files along with the unified *.str* file

More Advanced options

- `-b` or `--bond-order`: Discards input bond orders and forces assignment of bond orders based on the connectivity instead. This may NOT be useful for 3rd-row and heavier elements.
- `-e` or `--force-exhaustive`: Forces using "exhaustive algorithm" for finding new parameters. This is usually slower but might be faster in some situations.
- `-s` or `--server`: Makes the program run in "server mode", ie. a separate, self-contained output is generated for each input molecule.
- `-p` or `--parameter-file`: Reads in parameter files. Ordinarily, parameter files should be read from the rule file.
- `-l` or `--short-lone-pair`: Put halogen lonepair particles closer to the atom
- `-z` or `--keep-mol2-charge`: Keep charge values from Mol2 file

GROMACS-readable parameters

To get parameters in GROMACS format, execute the following command.

```
${SILCSBIODIR}/cgenff/cgenff_to_gmx mol=<mol2file>
```

This command produces three files <mol2file>_gmx.top, <mol2file>_gmx.pdb & <mol2file>_charmm.str. <mol2file>_gmx.top/pdb contain the GROMACS readable parameters and the coordinate information. <mol2file>_charmm.str contain CHARMM readable parameters.

Additionally, charmm36.ff directory containing the full CHARMM36 force field is given as output. This directory can be directly used to run simulations in GROMACS. For the ligand with filename ligand.mol2, the charmm36.ff directory will contain the following additional files:

- lig_ffbonded.itp : Contains bonded parameters for the ligand
- lig.rtp : Contains the residue topology for the ligand

11.3 CGenFF for Covalent Ligands

The CGenFF program can be used to parameterize modified amino acids covalently bound to ligands of interest. Currently, users can covalently link ligands to cysteine (CYS), lysine (LYS), tyrosine (TYR), serine (SER), threoine (THR), asparagine (ASN), and glutamate (GLU) amino acids. The resulting parameters, in CHARMM and GROMACS compatible formats, can be used to simulate a protein with a covalently bound ligand.

11.3.1 CGenFF for Covalent Ligands Using the CLI

SilcsBio provides the utility to assign topology and parameters to modified amino acids covalently bound to ligands of interest only in the command line interface (CLI). The following steps describe how to generate topology and parameters for modified amino acids as well as build protein structures mutated with the modified amino acids covalently bound to the ligand of interest.

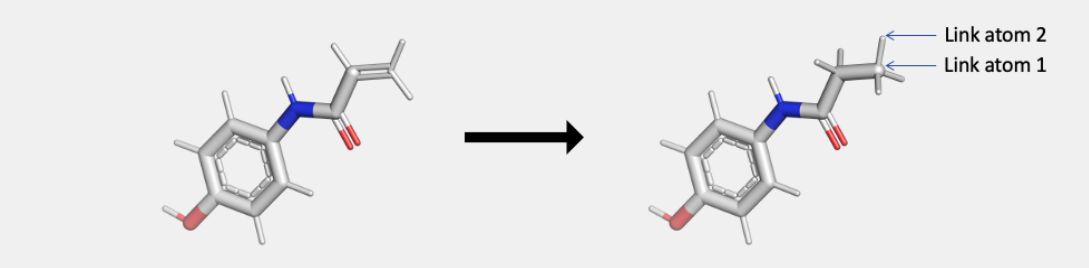
1. **Prepare the ligand Mol2:**

   The CGenFF program accepts Mol2 format files. For correct atom typing, it is important that the structure meets the following criteria:

   - all hydrogens are present
   - protonation and tautomeric state is correct
   - bond orders are correct

   The state of the ligand in the covalent bond should be considered when preparing the ligand Mol2 file. For example, the C=C double bond of an acrylamide molecule should

be changed to a C-C single bond for a covalently bonded acrylamide molecule. Additionally, the ligand must be docked into its binding pocket in the desired orientation with respect to the amino acid to which it will be covalently linked. If the user has a current SILCS-Small Molecule Suite license, then this may be accomplished using SILCS-MC docking, described in *SILCS-MC Docking Using the CLI* or *SILCS-MC Docking Using the SilcsBio GUI* of *SILCS-MC: Docking and Pose Refinement*, with the target residue selection corresponding to the covalently bound amino acid.

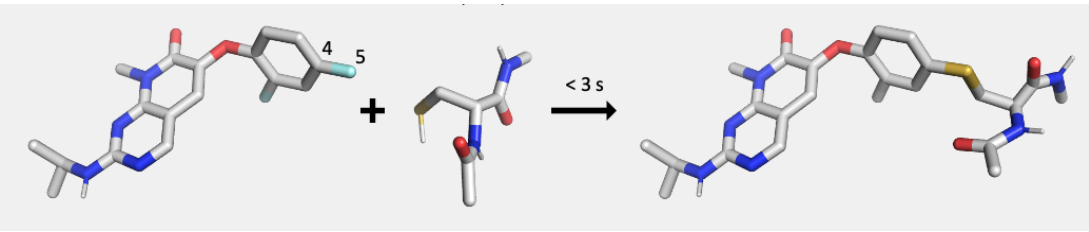


Once the Mol2 file is ready, note the atom ID for two atoms, Link atom 1 and Link atom 2 before you proceed to next step. Link atom 1 is the parent atom that will form a covalent bond with the amino acid side chain. Link atom 2 is the child atom that will get deleted once the bond is formed.

> **Warning:** The ligand Mol2 file must contain coordinates of the ligand within the desired binding pocket and in the proper orientation with respect to the amino acid to which the covalent bond will be formed. This can be conveniently acheived using SILCS-MC docking, described in *SILCS-MC Docking Using the CLI* or *SILCS-MC Docking Using the SilcsBio GUI* of *SILCS-MC: Docking and Pose Refinement*, with the target residue selection corresponding to the covalently bound amino acid.

2. **Build the modified amino acid covalently bound to the ligand:**

   Using the chemical group transformation module available through the SilcsBio software (*Chemical Group Transformations*), the ligand will be covalently linked to a desired amino acid (cysteine (CYS), lysine (LYS), tyrosine (TYR), serine (SER), threoine (THR), asparagine (ASN), or glutamate (GLU)). A new Mol2 file will be generated containing the energetically minimized, covalently bound amino acid.



Use the following command to generate the structure of the input ligand covalently bound to the desired amino acid:

```
$SILCSBIODIR/cgenff-covalent/1_build_mod_res lig=<ligand mol2␣
↪file>
linkatom1=<parent atom ID> linkatom2=<child atom ID>
linkresname=<linking amino acid>
```

Required parameters:

- The path and name of the input ligand Mol2 file:

```
lig=<location and name of ligand mol2>
```

- The atom ID (integer value) of the parent atom that will form the covalent bond with the amino acid side chain:

```
linkatom1=<atom ID for parent atom>
```

- The atom ID (integer value) of the child atom that will get deleted once the covalent bond to the amino acid side chain is formed.

```
linkatom2=<atom ID for child atom>
```

Note: Please ensure that the atom IDs entered for `linkatom1` and `linkatom2` correspond to the input Mol2 file entered for `lig`. The atom ID is listed in the first column of the Mol2 file under the `@<TRIPOS>ATOM` section.

- The residue name of the bound amino acid for covalent bonding with the ligand. The SilcsBio software supports covalent bonds to be formed with cysteine (CYS), lysine (LYS), tyrosine (TYR), serine (SER), threoine (THR), asparagine (ASN), and glutamate (GLU):

```
linkresname=<linking amino acid; CYS/LYS/ASN/SER/THR/
↪TYR/GLU>
```

Optional parameter:

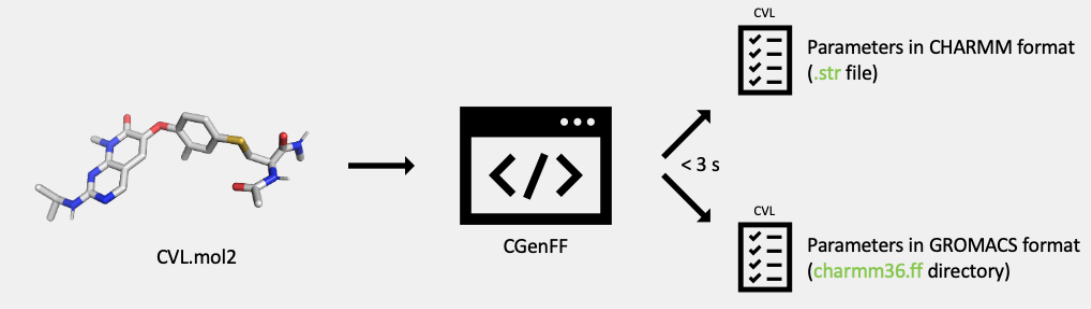
- The residue name and the basename of the output Mol2 file of the covalently bound amino acid. By default, `CVL` is used as the residue name:

```
newresname=<new residue name; default=CVL>
```

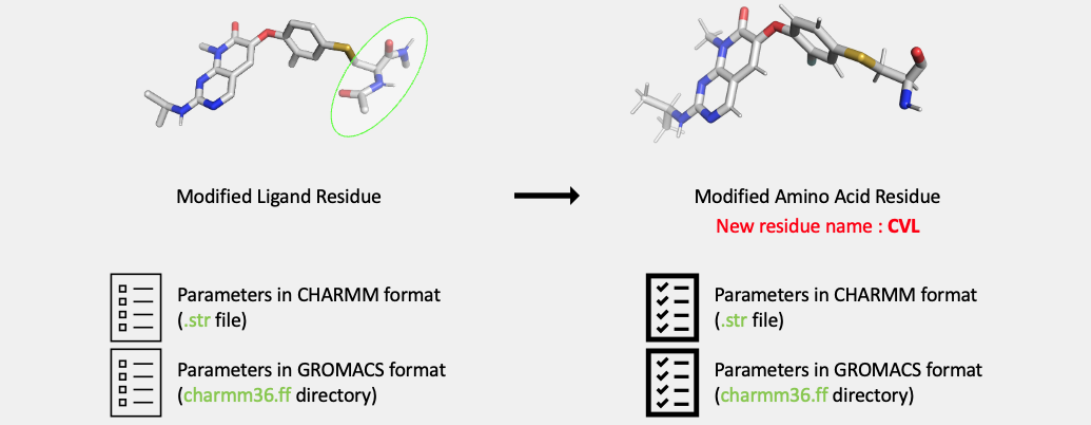
With the default `newresname`, `CVL`, the output from this step will be `cvl.mol2`.

3. **Generate the topology and parameters of the covalently bound amino acid:**

In this step, the CGenFF program generates the topology and parameters for the modified amino acid in Mol2 format created in the previous step. The topology and parameters for the modified amino acid are output in formats compatible with both the CHARMM and GROMACS MD simulation packages. The CHARMM compatible format will be output in a stream file (`.str` file). The GROMACS compatible format will be output in a directory (`charmm36.ff`).



The topology and parameters of the modified, covalently bound amino acid is converted to the topology and parameters of a residue that can be incorporated into a full protein sequence. The atom types of the backbone atoms of the modified amino acid are converted from CGenFF atom types to CHARMM36 atom types compatible with the CHARMM36m protein force field.



To generate the topology and parameters of the covalently bound amino acid, use the following command:

```
$SILCSBIODIR/cgenff-covalent/2_gen_rtp_pdb cvl=<covalently bound␣
↪amino acid mol2>
linkresname=<linking amino acid>
```

Required parameters:

- The path and name of the covalently bound amino acid Mol2 file. This file corresponds to the output Mol2 file from the previous step:

```
cvl=<location and name of the covalently bound amino␣
↪acid mol2>
```

- The residue name of the amino acid to which the ligand molecule is covalently bound. This entry corresponds to the same `linkresname` used in the previous step:

```
linkresname=<linking amino acid; CYS/LYS/ASN/SER/THR/
↪TYR/GLU>
```

Optional parameters:

- The residue number (integer value) of the covalently bound amino acid. The residue number of the covalently bound amino acid may be set to its corresponding residue number in the protein to which the ligand is covalently bound. This value can additionally be overwritten in the next step. The default value is set to 1:

```
linkresid=<linking amino acid resid; default=1>
```

- The chain ID of the covalently bound amino acid. The chain ID of the covalently bound amino acid can be defined in accordance with the target protein PDB:

```
linkchain=<linking amino acid chain ID; default=NULL>
```

With the default `newresname`, `CVL`, the outputs from this step will include the following:

- `cvl.pdb`
- `cvl.rtp`
- `toppar_cvl.str`
- `charmm36.ff`
- `toppar_all36_prot_modify_res.str`

4. **Mutate the protein:**

   The protein structure can now be modified to incorporate the covalently bound amino acid. In this step, the specified amino acid is mutated to the desired covalently bound amino acid in the input target protein PDB file. The structure, topology, and parameters of the covalently bound amino acid correspond to those generated in the previous steps.

Use the following command:

```
$SILCSBIODIR/cgenff-covalent/3_mutate_to_cvl prot=<protein PDB␣
↪file> cvl=<CVL PDB file>
linkresname=<linking amino acid> linkresid=<mutated residue ID>
```

Required parameters:

- The path and name of the target protein PDB file:

```
prot=<location and name of protein PDB file>
```

- The path and name of the covalently bound amino acid PDB file. This file corresponds to the output PDB file from the previous step:

```
cvl=<location and name of the covalently bound amino␣
↪acid pdb>
```

- The residue name of the amino acid to which the ligand molecule is covalently bound. This entry corresponds to the same `linkresname` used in the previous steps:

```
linkresname=<linking amino acid; CYS/LYS/ASN/SER/THR/
↪TYR/GLU>
```

- The residue number (integer value) of the covalently bound amino acid. The residue number of the covalently bound amino acid should be set to its corresponding residue number in the protein to which the ligand is covalently bound:

```
linkresid=<linking amino acid resid>
```

Optional parameters:

- The name of the output PDB file. The default name of the output pdb is set to `prot-cvl.pdb`:

```
outputpdb=<name of the output PDB file; default="prot-
↪cvl.pdb">
```

Tip: You may wish to try the example provided in `${SILCSBIODIR}/examples/cgenff-covalent/` to familiarize yourself with the CGenFF for Covalent Ligands workflow.

The outputs of the CGenFF for Covalent Ligands workflow may be used for *Standard Molecular Dynamics (MD) Simulations* or, with the SILCS-Small Molecule Suite, *SILCS Simulations*. For these simulations, the initial structure will correspond with the `outputpdb`, `prot-cvl.pdb` by default.

- To perform standard MD simulations of a protein with a covalent ligand, follow the instructions provided in *Standard Molecular Dynamics (MD) Simulations* with `covalent=true` specified for the first, `${SILCSBIODIR}/md/1_setup_md_boxes` step. Specify the `outputpdb` from this section (`prot-cvl.pdb` by default) as the initial structure with `prot=prot-cvl.pdb`.
- To perform SILCS simulations of a protein with a covalent ligand, the user must have a current license for the SILCS-Small Molecule Suite. Please contact info@silcsbio.com for more information. To set up the SILCS simulations, follow the instructions provided in *SILCS Simulations Using the CLI*. In the first, `${SILCSBIODIR}/silcs/1_setup_silcs_boxes` step, specify the `outputpdb` from this section (`prot-cvl.pdb` by default) as the initial structure with `prot=prot-cvl.pdb`. This will produce an error due to the `CVL` residue being unrecognized. In the resulting `1_setup` directory, replace the `charmm36.ff` directory with the GROMACS formatted output from CGenFF-covalent, also named `charmm36.ff`. Continue the setup process by repeating the first, `${SILCSBIODIR}/silcs/1_setup_silcs_boxes` step with `skip_pdb2gmx=true`. Continue to the following steps as described in *SILCS Simulations Using the CLI*. Note that it is typically NOT recommended to run a SILCS simulation with any ligand bound in the binding pocket.

For simulations with the CHARMM program, the `outputpdb`, `prot-cvl.pdb` by default, will need to be edited to fit CHARMM PDB format (e.g., adding the segname column) and the output topology and parameter stream files, `toppar_cvl.str` and `toppar_all36_prot_modify_res.str` by default, should be added to the CHARMM input.

11.4 Standard Molecular Dynamics (MD) Simulations

Standard MD simulations are useful tools that act as a "computational microscope" for understanding the dynamic structure and function of biomolecular systems. Through MD simulations, static structures become time-evolved snapshots of how molecules behave and interact over time. Structural and energetic analyses can be combined to describe the MD simulated biomolecular systems, including, but not limited to, the determination of protein stability and identifying key interactions promoting ligand activity. SilcsBio offers the utilities needed to set up, run, and analyze proteins and protein–ligand complexes. The analyses available through the standard MD simulation utility of the CGenFF Suite are:

- MD simulations of protein, ligand, or protein–ligand complex (*MD Simulations of a Protein and Ligand*)
  - Protein backbone root mean squared deviation (RMSD)
  - Per-residue root mean squared fluctuation (RMSF)
  - Protein radius of gyration
  - Protein solvent accessible surface area (SASA)
  - Ramachandran plot
  - Ligand RMSD
  - Ligand RMSF
  - Ligand center-of-mass (COM)
  - Protein–ligand interaction energy
  - Ligand SASA
- MD simulations of a membrane protein or membrane protein–ligand complex (*MD Simulation for Membrane Proteins*)
  - Protein backbone root mean squared deviation (RMSD)
  - Per-residue root mean squared fluctuation (RMSF)
  - Protein radius of gyration
  - Protein solvent accessible surface area (SASA)
  - Ramachandran plot
  - Ligand RMSD
  - Ligand RMSF
  - Ligand center-of-mass (COM)
  - Protein–ligand interaction energy
  - Ligand SASA
- MD simulations of a protein in the presence of probe solutes (*MD Simulation with Probes*)
  - Protein backbone root mean squared deviation (RMSD)
  - Per-residue root mean squared fluctuation (RMSF)
  - Protein radius of gyration
  - Protein solvent accessible surface area (SASA)
  - Ramachandran plot
  - Protein pocket identification

In addition to the above listed analyses, users may also apply their own, custom analyses to the simulation trajectories output from the standard MD simulation utility of the CGenFF Suite.

11.4.1 MD Simulations of a Protein and Ligand

MD Simulations Using the CLI

SilcsBio provides command line interface (CLI) utilities to set up, run, and analyze standard MD simulations. The following steps describe how to set up, run, analyze, and clean standard MD simulations of a protein, ligand, or protein–ligand complex.

1. **Set up the MD simulation system:**

   To set up a standard MD simulation with a protein target, use the following command:

   ```
   ${SILCSBIODIR}/md/1_setup_md_boxes prot=<Protein PDB>
   ```

   Required parameter:

   - Path and name of the input protein PDB file:

     ```
     prot=<protein PDB file>
     ```

   Warning: The setup program internally uses the GROMACS utility `pdb2gmx`. You may encounter errors while processing the protein PDB file at this stage. The most common reasons for errors at this stage involve mismatches between the expected residue names or atom names in the input PDB file and those defined in the CHARMM36 force-field.

   To fix this problem, run the `pdb2gmx` command manually from within the `1_setup` directory to get a detailed error message. The setup script already prints out the exact commands you need to run to see this message. The commands are also shown below:

   ```
   cd 1_setup
   pdb2gmx -f <PROT PDB NAME>_4pdb2gmx.pdb -ff charmm36 -water␣
   ↪tip3p -o test.pdb -p test.top -ter -merge all
   ```

   Once all the errors in the input PDB are corrected, rerun the setup script.

   By default, 1 independent simulation system will be created. The simulation system will comprise your protein or protein–ligand structure solvated in a waterbox neutralized by addition of counter ions. The size of the waterbox will be set such that there is a 10 angstrom margin on all sides of the protein.

The setup command offers a number of options as shown below. The full list of options can be reviewed by simply entering the command without any options.

Optional parameters:

- Number of independent simulation systems:

```
numsys=<# of simulations; default=1>
```

  If additional independent simulation systems are needed, specify the number of systems with the `numsys` keyword. For example, if 6 simulation systems are needed, include `numsys=6` in the command.

- Path and name of non-covalent ligand/cofactor mol2 files:

```
lig=<ligand MOL2 files; e.g. "lig1.mol2,/home/johndoe/
↪lig2.mol2"; default=empty/NULL>
```

  If the input structure contains ligands/cofactors and you wish to run MD simulations of the protein–ligand complex, provide the path and name of each ligand Mol2 file separated by comma using the `lig` keyword. The ligand poses must be properly oriented with respect to the target protein. For covalent ligands, see additional parameters below.

  For users with access to the SILCS-Small Molecule Suite, if the pose of the ligand originates from SILCS-MC Docking or Pose Refinement, please refer to the *Best Protein–Ligand Complex Retrieval* for further instructions.

Tip: If you want to run MD with only your non-covalent ligands then use `prot=none`.

- Application of weak harmonic positional restraints:

```
restraints_mode=<level of restraints on protein; none/
↪flex/calpha/tight/rigid/custom; default=none>
```

  A weak harmonic positional restraint, with a force constant of ~0.12 kcal/mol/Å^2, can be applied to all C-alpha atoms during MD simulations. The `restraints_mode` keyword allows the user to customize the restraints applied to the protein. By default, no restraints are applied.

  There are five options available:

  - `flex` *Selection*: C-alpha atoms of begin and end residues of HELIX and SHEET entries in the PDB *Force Constant*: 50 kJ/mol/nm^2(~0.12 kcal/mol/Å^2)

  - `calpha` *Selection*: C-alpha atoms of all protein residues *Force Constant*: 50 kJ/mol/nm^2
  - `tight` *Selection*: Backbone atoms N, CA and C of all protein residues *Force Constant*: 100 kJ/mol/nm^2
  - `rigid` *Selection*: Non-hydrogen atoms of all residues *Force Constant*: 150 kJ/mol/nm^{2}By default, using the `rigid` option will turn off the side chain scrambling.
  - `custom` *Selection*: Custom selection of atoms provided as "restraints_string=" in the command line *Force Constant*: Custom force constant provided as "restraints_fc=" in the command line For example, if `restraints_string="r 17-160 | r 168-174 & a CA"` and `restraints_fc=1000`, then a force constant of 1000 kJ/mol/nm^2 will be applied to the C-alpha atoms in residues 17 to 160 and residues 168 to 174. Note that the custom selection string should follow the GROMACS selection syntax.

- Reorientation of residue side chains:

```
scramblesc=<true/false; default=false>
```

  When set to `true` the `scramblesc` keyword reorients the side chain dihedrals of the protein residues.

- Simulation box size:

```
margin=<system margin (Å); default=10>
```

  By default, the simulation system size is determined by adding a 10 Å margin to the size of the input protein structure. The size of the simulation box can be customized by using the `margin` keyword.

Tip: A larger margin size than the default 10 Å is recommended if the conformation of the protein is expected to change during the simulation.

Warning: Do not use a margin size smaller than 10 Å. A smaller margin size will increase the likelihood of artifacts due to "self" interaction across the periodic boundary conditions.

- Concentration of ions:

```
ions_conc=<concentration of salt (M); default=0>
```

The concentration of the ions in the simulation system can be specified in M units with the `ions_conc` keyword. The number of ions is calculated based on the number of water molecules in the simulation system and the the user-defined ion concentration relative to 55 M water. By default, the concentration of ions is set to `0`. Note that, if an ion concentration is specified, these ions will be added in addition to neutralizing ions. Additionally, note that neutralizing ions will be added even if the `ions_conc=0` if the protein or protein–ligand structure is charged.

- Type of ions added to the simulation system:

```
ions_type=<type of salt; options: KCl, NaCl;␣
↪default=KCl>
```

The cation included in the simulation system can be specified to be either Na or K using the `ions_type` keyword. For example, add `ions_type=NaCl` if you want to add sodium chloride ions to the simulation system. The default value is *KCl*.

- Skip the GROMACS utility `pdb2gmx`:

```
skip_pdb2gmx=<true/false; default=false>
```

If the `pdb2gmx` utility has already been successfully run, you may skip rerunning this utility with the `skip_pdb2gmx` keyword.

Covalent Ligand Parameters:

- Add covalent ligands to the system:

```
covalent=<true/false; default=false>
```

If the input structure contains covalent ligands and you wish to run MD simulations in the presence of the ligands, then extract each ligand from the input PDB file and save as a separate Mol2 file with the ligand in the correct state, position and orientation with respect to the target protein.

See *CGenFF for Covalent Ligands Using the CLI* for more information on preparing covalent ligands for MD simulations.

Note: The covalent ligands must not be included in the `lig` keyword. Only non-covalent ligands should be included in the `lig` keyword.

- When covalent=true, provide the path and name of the covalent input file:

```
covalentinp=<covalent input file; default=covalent.inp>
```

Covalent input file format (5 columns separated by space):

<lig-file-path> <linkatom1> <linkatom2> <linkresname> <linkresid>

Where:

- <lig-file-path> = ligand MOL2 file path; e.g., /home/johndoe/lig2.mol2
- <linkatom1> = atom ID for parent atom (integer) [typically a non-hydrogen atom]
- <linkatom2> = atom ID for child atom (integer) [typically a hydrogen atom]
- <linkresname> = residue name for covalent link; CYS/LYS/ASN/SER/THR
- <linkresid> = residue ID for covalent link (integer)

1. **Run the MD simulation:**

   To perform MD simulations, use the following command:

```
${SILCSBIODIR}/md/2a_run_md prot=<Protein PDB> cycles=<number of␣
↪cycles to run>
```

   The command to run MD simulations offers a number of options as shown below. The full list of options can be reviewed by entering the command without any options.

   Optional parameters:

   - Duration of the MD simulation:

```
cycles=<# of production cycles with 10 ns/cycle;␣
↪default=50>
```

     By default, the MD simulations are run in continuous cycles of 10 ns. The duration of the simulation can be specified using the `cycles` keyword. The default value is set to 50, which will correspond to a continuous 500 ns simulation split into 50 trajectory files of 10 ns each.

   - Temperature of the MD simulations:

```
temp=<temperature of simulation; default=310>
```

     By default the temperature of the simulation system is set to 310 K. The temperature of the simulation system can be specified with the `temp` keyword. For example, use `temp=300` to set the temperature of the simulation system to 300 K.

   - Manually submit simulation jobs to the queuing system:

```
batch=<true/false, true: generate files but do not␣
↪submit jobs; default=false>
```

To generate the simulation input files without submitting the simulation jobs, set the `batch` keyword to `batch=true`. This keyword may be benefitial to those who wish to modify the input files before performing the simulations.

- Number of cores per simulation:

```
nproc=<# of cores per simulation; default nproc=8>
```

If the MD simulation will be performed using only CPUs, then the number of CPUs used to run the MD simulation can be specified using the `nproc` keyword. For GPU+CPU nodes, GROMACS performs best using the default settings (`nproc=8`). Set `nproc` if less than 8 CPUs are available for the simulation.

- Continue the simulation from a previous cycle

```
restart=<true/false, true:- restart all runs from␣
↪last cycle; default=false>
```

Set the `restart` keyword to `restart=true` if you want to restart all the simulation runs from the last cycle. To restart a specific run, the run can be specified. For example, if only simulation number 2 should be restarted, `restart=#2` should be specified.

3. **Collect the simulation trajectory:**

The resulting trajectory can be collected by using the following command.

```
${SILCSBIODIR}/md/2b_trjcat prot=<Protein PDB>
```

The above command will first create a concatenated trajectory for each independent simulation system. Each trajectory will be saved as `2b_trajcat/traj_X_fit.xtc` file with *X* indicating the system number. In the case that only one simulation system was run, there will only be a `2b_trajcat/traj_1_fit.xtc`.

Once the trajectory from independent simulations are concatenated, a single trajectory is created that concatenates the protein trajectories created from independent simulations for use in analysis. The trajectory will be translated and rotated to fit to the first frame of the trajectory. `traj.pdb` and `traj.xtc` will be your final output files.

Optional parameters:

- Number of independent simulation systems:

```
numsys=<# of simulations; default based on 2a_run_md␣
↪step>
```

If additional independent simulation systems need to be collected, specify the number of systems with the `numsys` keyword. For example, if 6 simulation systems are to be collected, include `numsys=6` in the command. The default value

is based on the `numsys` specified previously with `2a_run_md`.

- Number of cycles (Duration of the MD simulation):

```
cycles=<# of production cycles with 10 ns/cycle;␣
↪default based on 2a_run_md step>
```

To change the number of cycles from each independent simulation systems to collect, specify the `cycles` keyword with the number of desired cycles. The default value is based on the `cycles` specified previously with `2a_run_md`.

- Periodic Boundary Condition:

```
pbctype=<type of periodic boundary condition␣
↪treatment; res/atom/nojump/cluster/whole;␣
↪default=whole>
```

By default, the trajectory will be collected with periodic boundary condition to make molecules whole. The periodic boundary conditions used to collect the simulation trajectories can be specified using the `pbctype` keyword.

Tip: For simulations of protein–ligand complex structures in which the ligand is not covalently bound to the protien, using `pbctype=nojump` may be benefitial. For simulations of multi-domain proteins, using `pbctype=whole` may be beneficial. Further processing of the trajectory can be performed using the `trjconv` utility from GROMACS if desired using the following commands:

```
cp traj.xtc traj_bak.xtc
$GMXDIR/gmx trjconv -s ref.tpr -f traj_bak.xtc -pbc $
↪{pbctype} -center -o traj.xtc
```

Additional collection parameters:

- Keep solvent molecules

```
full=<true/false, true:- collect with all atoms;␣
↪default=false>
```

By default, the trajectories will be stripped of all solvent molecules. To retain the solvent molecules in the trajectories, set the `full` keyword to `full=true`.

- Strip hydrogens from the simulation trajectory:

```
nohyd=<true/false, true: collect with all atoms␣
↪except hydrogens; default=false>
```

By default, hydrogen atoms are retained when the simulation trajectory is collected. To strip hydrogens from the trajectories, set the `nohyd` keyword to `nohyd=true`.

4. **Perform basic trajectory analyses:**

   The `traj.xtc` trajectory file will be analyzed for a number of common properties such as RMSD, RMSF, RGYR and interaction energies.

   ```
   ${SILCSBIODIR}/md/3_analyze prot=<Protein PDB>
   ```

   The command will perform the following analyses using GROMACS:

   - Backbone root mean squared deviation (RMSD):

     ```
     rmsdbb=<calc rmsd of protein backbone; true/false;␣
     ↪default=true>
     ```

     Calculate the RMSD of the protein backbone. The resulting values are in units.

   - Per-residue root mean squared fluctuation (RMSF):

     ```
     rmsfprot=<calc rmsf of protein per residue; true/
     ↪false; default=true>
     ```

     Calculate the per-residue RMSF of the simulated protein. The resulting values are in units.

   - Protein radius of gyration (RGYR):

     ```
     rgyr=<calc radius of gyration of protein; true/false;␣
     ↪default=true>
     ```

     Calculate the RGYR of the simulated protein. The resulting values are in units.

   - Protein solvent accessible surface area (SASA):

     ```
     sasaprot=<sasa of protein residues; e.g., "1" or "r 1-
     ↪9 | r 16-24 & 1"; default=none>
     ```

     Calculate the SASA of the simulated protein. The selection terms defined in the index groups generated by the `gmx make_ndx` command can be used to specify which residues to calculate SASA. For example, *sasaprot=1* will select all protein residues. The resulting values are in units.

   - Ramachandran plot for Phi-Psi dihedrals:

     ```
     phipsi=<calc phi-psi angles and get Ramachandran plot;
     ↪ "all" or "2-9,34-66"; default=false>
     ```

Calculate the phi-psi dihedral distributuion of the simulated protein and generate a Ramachandran plot. Default value is `false`. To consider all residues, use `phipsi=all`. To consider only specific residues, use `,` and `-` to define the range of residues. For example, `phipsi=2-9,34-66` will consider residues 2 to 9 and 34 to 66.

If a **ligand** is present, the following analyses will be performed:

- Ligand root mean squared deviation (RMSD)

```
rmsdlig=<calc rmsd of ligand; true/false;␣
↪default=true>
```

Calculate the RMSD of the simulated ligand. The resulting values are in units.

- Ligand root mean squared fluctuation (RMSF)

```
rmsflig=<calc rmsf of ligand residue; true/false;␣
↪default=true>
```

Calculate the per-residue RMSF of the simulated ligand. The resulting values are in units.

- Center-of-mass (COM) of the ligand:

```
comlig=<calc center of mass of ligand; true/false;␣
↪default=true>
```

Calculate the COM the simulated ligand. The resulting values are in units.

- Interaction energy between the protein and ligand:

```
IEprotlig=<calc interaction energy between protein␣
↪and ligand; true/false; default=false>
```

Calculate the interaction energy between the protein and the ligand. The resulting values are in units.

Note: The interaction energy represents the strength of interaction between two molecules (in vacuum) and does not account for polar interaction solvation energy. Thus, the interaction energy should not be confused with binding affinity. Interaction energy and its decomposition into van der Waals and electrostatic components can give insights into the driving forces of protein–ligand binding.

- Solvent accessible surface area (SASA) of the simulated ligand:

```
sasalig=<calc sasa of ligand; true/false;␣
↪default=false>
```

Additional Parameters:

- Analysis output directory:

  ```
  anlyzdir=<name of analysis output directory;␣
  ↪default=3_analyze>
  ```

  The output data and plots will be collected in the `3_analyze` directory (or the user-specified directory). This directory can be downloaded to a local machine for visualization and analysis.

- Python executable path:

  ```
  python=<path to python executable for plotting;␣
  ↪default=>
  ```

  This script uses the *Matplotlib* python package for plotting the data. The path to the python executable with the matplotlib module installed must be specified. The default python path can be checked using the which python command. Please refer to *Python 3 Requirement* for more information.

Note: The plots use `boxplots` to show the distribution of data. The boxplot is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum. More information can be found at https://en.wikipedia.org/wiki/Box_plot

---

5. **Clean and compress MD data:**

   After performing the analyses the output data may be cleaned to save disk space by removing unnecessary files and compressing the trajectories. SilcsBio provides the `4_cleanup` utility to facilitate the data cleanup:

   ```
   ${SILCSBIODIR}/md/4_cleanup prot=<Protein PDB>
   ```

   The command `4_cleanup` will delete unnecessary files in the *2a_run_md* directory and compress the raw trajectory data. Individual run directories can be deleted by re-running the command with the `delete=true` keyword included; the default value is `delete=false`:

   ```
   ${SILCSBIODIR}/md/4_cleanup prot=<Protein PDB> delete=true
   ```

11.4.2 MD Simulation for Membrane Proteins

MD Simulations for Membrane Proteins Using the CLI

SilcsBio provides the utility to run standard MD simulations of membrane proteins only in the command line interface (CLI). The following steps describe how to set up, run, analyze, and clean standard MD simulations of a membrane protein.

1. **Set up the MD simulation system:**

   To set up a standard MD simulation with a membrane protein target, you may use either a bare protein structure or your own pre-built protein/bilayer system with MD module.

   If you are beginning with a bare protein, SilcsBio provides a command line utility to embed transmembrane proteins, such as GPCRs, in a bilayer membrane for subsequent MD simulations:

```
${SILCSBIODIR}/md/memb/1a_fit_protein_in_bilayer prot=<Protein␣
↪PDB>
```

   By default, the protein will be embeded in a bilayer of 9:1 POPC/cholesterol. Users may also choose from four bilayer compositions using the `bilayer` option. The available bilayer compositions are pure POPC (100% POPC) (`bilayer=1`), 9:1 POPC/cholesterol (`bilayer=2`), 8:2 POPC/cholesterol (`bilayer=3`), and 7:3 POPC/cholesterol (`bilayer=4`). Alternatively, users may provide their own bilayer structure using the `bilayerpdb` option.

   The `1a_fit_protein_in_bilayer` command will align the first principal axis of the protein with the bilayer normal (Z-axis) and translate the protein center of mass to the center of the bilayer (Z=0).

   If the protein is already oriented as desired, `orient_principal_axis=false` will suppress automatic principal axis-based alignment.

```
${SILCSBIODIR}/md/memb/1a_fit_protein_in_bilayer prot=<Protein␣
↪PDB> orient_principal_axis=false
```

   By default the system size is set to 120 Å along the X and Y dimensions. To change the system size, use the `bilayer_x_size` and `bilayer_y_size` options.

```
${SILCSBIODIR}/md/memb/1a_fit_protein_in_bilayer prot=<Protein␣
↪PDB> bilayer_x_size=<X dimension> bilayer_y_size=<Y dimension>
```

   By default, the script fits the bare protein into the lipid bilayer such that the center of mass of the protein is aligned with the center of mass of the bilayer. This may not be the best choice for your membrane proteins, and you may therefore want to adjust the position of your protein according to a recommendation from an external program such as the OPM server. In this case you can adjust the relative position of the protein

with respect to the position of the bilayer by using the `offset_z` option to specify the distance by which the center of mass of the protein should be offset in Z-direction from the center of the bilayer.

```
${SILCSBIODIR}/md/memb/1a_fit_protein_in_bilayer prot=<Protein␣
↪PDB> offset_z=<distance_in_Z-direction>
```

Note: To use the OPM server (https://opm.phar.umich.edu/ppm_server) output as a guide to build a transmembrane system for MD simulations, please see *How do I fit my membrane protein in a bilayer as suggested by the OPM server?*

The resulting protein/bilayer system will be output in a PDB file with suffix `_popc_chol`. It is important to use molecular visualization software at this stage to confirm correct orientation of the protein and size of the bilayer before using this output as the input in the next step.

Alternatively, you may use a protein/bilayer system that has been built using other software tools.

The following command will prepare the SILCS simulation box by adding water and neutralizing ions to the protein/bilayer system:

```
${SILCSBIODIR}/md/memb/1b_setup_md_with_prot_bilayer prot=
↪<Protein/bilayer PDB>
```

By default, 1 independent simulation system will be created. The setup command offers a number of options. The full list of options can be reviewed by simply entering the command without any options. The options are same as described in step 1 for MD simulations of globular proteins *MD Simulations Using the CLI*.

Tip: You may use `batch=false` option to run this step on your head node. Note that it can sometimes take few hours to finish this step for very large systems if your head node has slow CPUs and/or limited memory and/or many users are actively using the head node.

---

1. **Run the MD simulation:**

   To perform MD simulations, use the following command.

   ```
   ${SILCSBIODIR}/md/2a_run_md prot=<Protein/bilayer PDB> cycles=
   ↪<number of cycles to run>
   ```

   By default, this command detects `memb=true` as this is membrane protein.

   For other parameters, see step 2 of *MD Simulations Using the CLI*

3. **Collect the simulation trajectory:**

   The resulting trajectory can be collected by using the following command.

   ```
   ${SILCSBIODIR}/md/2b_trjcat prot=<Protein/bilayer PDB>
   ```

   For parameters, see step 3 of *MD Simulations Using the CLI*

4. **Perform basic trajectory analyses:**

   The `traj.xtc` trajectory file will be analyzed for RMSD, RMSF, RGYR, Interaction energies and SASA.

   ```
   ${SILCSBIODIR}/md/3_analyze prot=<Protein/bilayer PDB>
   ```

   For parameters, see step 4 of *MD Simulations Using the CLI*

   The output data and plots will be collected in the `3_analyze` directory (or the user-specified directory). This directory can be downloaded to a local machine for visualization and analysis.

5. **Clean and compress MD data:**

   After the analyses you can clean the output data for unnecessary files and compress the trajectories.

   ```
   ${SILCSBIODIR}/md/4_cleanup prot=<Protein/bilayer PDB>
   ```

   The command will delete the un-necessary files in `2a_run_md` directory and compress the raw trajectory data for you.

   You can then delete the individual run directories by re-running the command with `delete=true`. Default value is `false`.

   ```
   ${SILCSBIODIR}/md/4_cleanup prot=<Protein/bilayer PDB>␣
   ↪delete=true
   ```

11.4.3 MD Simulation with Probes

Background

MD simulations encompassing proteins in the presence of organic probes sample the interactions between small functional groups and proteins and are widely used in various computer-aided drug discovery methods.

The addition of hydrophobic probes in MD simulations can open and stabilize the openings of cryptic pockets that may otherwise collapse during a simulation. Cryptic pockets are a type of binding pocket that are typically not apparent in a crystal structure, but can be induced to open

when a ligand is nearby. It is hypothesized that the hydrophobic side chains that often encompass the cryptic pocket and surround the opening of the pocket must undergo conformational changes to fully open the cryptic pocket. Such conformational changes can be facilitated in MD simulations through the inclusion of apolar organic solutes such as benzene or propane.

MD-with-Probe Simulations Using the CLI

SilcsBio provides the utility to run MD simulations of proteins in the presence of probe molecules. The following steps describe how perform MD simulations with probes using the command line interface (CLI).

1. **Set up the MD-with-probe simulation:**

   To set up a standard MD simulation with a protein target and probe molecules, use the following command.

   ```
   ${SILCSBIODIR}/md/probe/1_setup_md_boxes prot=<Protein PDB>␣
   ↪probe=<probe to be added>
   ```

   By default, 6 independent simulation systems will be created with 10 angstrom margin on all sides and neutralized by addition of counter ions.

   The setup command offers same options as shown in step 1 of *MD Simulations Using the CLI* with additional options shown below. The full list of options can be reviewed by simply entering the command without any options.

   - The probe molecule. By default, `benx` (benzene) will be used as probe. SilcsBio also supports the use of `prpx` (propane), `form` (formamide), `dmee` (dimethylether), `meoh` (methanol), and `imia` (imidazole):

     ```
     probe=<probe to be added; default=benx>
     ```

   - Concentration of the probe (in M). The concentration of the probe within the simulation system can be specified using the `conc` keyword followed by the desired concentration in M. For example, add `conc=0.25` for a probe concentration of `0.25` M. The default value is `1`:

     ```
     conc=<concentration of probe; default=1 M>
     ```

   - Different concentration of the probe across independent simulations (in M). For MD simulations of a protein in the presence of different concentration of the probe, the `diffconc=true` flag will prompt the SilcsBio software to assign different concentrations to each simulation system. For example, including `diffconc=true` in conjunction with `probe=benx`, `numsys=6`, and `conc="0 0.2 0.4 0.6 0.8 1"` will result in 6 independent simulation systems of the protein in (1) the absence of benzene probes, and in the presence of (2) 0.2 M ben-

zene, (3) 0.4 M benzene, (4) 0.6 M benzene, (5) 0.8 M benzene, and (6) 1 M benzene.

```
diffconc=<true/false, true: numsys=6 & conc="0 0.2 0.4 0.6 0.
↪8 1"; default=false>
```

2. **Run the MD simulation:**

   To perform MD simulations, use the following command.

   ```
   ${SILCSBIODIR}/md/2a_run_md prot=<Protein PDB> cycles=<number of␣
   ↪cycles to run>
   ```

   Optional parameter:

   - Duration of the MD simulation:

     ```
     cycles=<# of production cycles with 10 ns/cycle;␣
     ↪default=20>
     ```

     By default, the MD simulations are run in continuous cycles of 10 ns. The duration of the simulation can be specified using the `cycles` keyword. The default value is set to 20, which will correspond to a continuous 200 ns simulation split into 20 trajectory files of 10 ns each.

   For other parameters, see step 2 of *MD Simulations Using the CLI*. All parameters can also be viewed by entering `${SILCSBIODIR}/md/2a_run_md` with no inputs.

3. **Collect the simulation trajectory:**

   The resulting MD simulation trajectory can be collected by using the following command.

   ```
   ${SILCSBIODIR}/md/2b_trjcat prot=<Protein PDB>
   ```

   For parameters, see step 3 of *MD Simulations Using the CLI*. The parameters can also be viewed by entering `${SILCSBIODIR}/md/2b_trjcat` with no inputs.

4. **Perform basic trajectory analyses:**

   The `traj.xtc` trajectory file will be analyzed to output the root mean squared deviation (RMSD) of the protein, per-residue root mean squared fluctuation (RMSF) of the protein, radius of gyration (RGYR) of the protein, interaction energies of the protein to the probe molecules, and solvent accessible surface area (SASA) of the protein.

   ```
   ${SILCSBIODIR}/md/3_analyze prot=<Protein PDB>
   ```

   For information on the parameters and available trajectory analyses, see step 4 of *MD Simulations Using the CLI*. All available parameters and trajectory analysis can also be viewed by entering `${SILCSBIODIR}/md/3_analyze` with no inputs.

The output data and plots will be collected in the `3_analyze` directory (or the user-specified directory). This directory can be downloaded to a local machine for visualization and analysis.

5. **Identify pockets:**

   To quickly identify a pocket, use the `mdpocket` tool that comes with an open source project `fpocket`.

```
mdpocket --trajectory_file traj.xtc --trajectory_format xtc -f␣
↪traj.pdb
```

   The above command creates several files, however the output file, `mdpout_dense_grid.dx`, will contain all data necessary for identifying pockets. The `mdpout_dense_grid.dx` file contains the probability of an open cavity existing in each voxel. This information can conveniently be visualized using PyMOL or VMD.

   For PyMOL, use the following commands:

```
load traj.pdb
load mdpout_dens_grid.dx
isosurface pocket, mdpout_dens_grid, <cutoff>
```

   The higher the `<cutoff>` value is the volume becomes smaller as those voxels are more open during the simulation. Typically using `2` or `3` should give more transient pockets and `5` or `6` should give more open pocket.

6. **Clean and compress MD data:**

   To clean and compress MD data, see step 5 of *MD Simulations Using the CLI*.

11.5 CGenFF Parameter Optimizer

The CGenFF program performs atom typing and assignment of parameters and charges by analogy in a fully-automated fashion. Atom typing is done by a deterministic programmable decision tree. Assignment of bonded parameters is based on substituting atom types in the definition of the desired parameter. A penalty is associated with every substitution and the existing parameter in the current version of the force field with lowest total penalty is chosen as an approximation for the desired parameter. The penalty score is returned to the user as an indication of the approximation needed to obtain the desired parameter. For example, dihedral parameters with higher penalties (typically >50) may be candidates for further optimization. When using the Optimizer, the atom typing and parameter assignment is performed internally such that the user only needs to supply the Mol2 file of the molecule of interest.

The threshold for selecting parameters for optimization is based on multiple criteria, including user preference. The penalty score is determined by the analogy of the atom types associated with the

new parameter to those of parameters already in the CGenFF force field. Parameters with high penalties may actually be of satisfactory accuracy while parameters with low penalties may be of poor accuracy. Accordingly, the user should make an initial evaluation of the treatment of a given molecule by CGenFF and then identify the parameters that may contribute to possible inaccuracies in the conformational properties of the molecule or its interactions with the environment. Dihedral parameters will typically have the largest impact on the conformational properties of a molecule and, therefore, should be considered for optimization.

CGenFF Parameter Optimizer allows for automatic optimization of rotatable dihedrals. Once the user specifies the dihedral to be optimized, the Optimizer coordinates the generation of quantum mechanical (QM) target data followed by the fitting of force field parameters to these target data. The Optimizer will first spawn Psi4 QM jobs. It will then collate the resulting QM dihedral scan data. Finally, it will fit force field parameters to these data using the least-square fitting procedure LSFitPar and output the new, optimized force field parameters. If multiple dihedrals are to be optimized, then each dihedral is fitted separately targeting independent QM scans on each dihedral. During the fitting, the initial multiplicities are those assigned by the CGenFF program. Once the initial fit is complete and the RMS error determined, the program automatically refits the data using a multiplicity of 1, then multiplicities of 1 and 2, 1, 2 and 3 and 1, 2, 3 and 6. If improvements in the RMSE are obtained beyond a cutoff (default is 10% of the RMSE) with the additional multiplicities, then those parameters are selected.

The Optimizer accepts molecules in Mol2 file format, with the following requirements: hydrogens must be explicitly defined, ionizable groups must have correct protonation states, and bond-order between atoms must be correctly defined.

To run the program, set the following environment variables,

```
export SILCSBIODIR=<silcsbio>
export PSI4DIR=<psi4>
export GMXDIR=<gromacs/bin>
```

The Psi4 package can be obtained from the Psi4 download page. For the easiest installation, use the Psi4 Binary Installer, run the command:

```
./Psi4conda-latest-py36-Linux-x86_64.sh
```

and following the step-by-step guide.

After Psi4 has been installed, set the `PSI4DIR` variable correctly and proceed with using the optimizer. To make sure `PSI4DIR` is correctly set, check if `$PSI4DIR/bin/psi4` is accessible.

Note that downloading the Psi4 source code and compiling the program locally with the appropriate switches can lead to a significant gain in performance over the downloaded binary version.

11.5.1 Usage

```
${SILCSBIODIR}/cgenff-optimizer/optimize_cgenff_par mol=<mol2file>
```

The second line of the Mol2 file contains a string naming the molecule. Certain molecular modeling programs will put the string "*" on this line instead of a descriptive name for the molecule. In such cases, please replace the string with an alphanumeric string, else the Optimizer will fail.

Along with the required argument of `mol=<mol2file>`, the following additional parameters can also be set:

1. Penalty score :

   ```
   penalty=<score; default 50>
   ```

   By default, the program identifies rotatable dihedrals with penalties greater than 50 for optimization. This can be changed by adjusting the `penalty` argument, to increase/decrease the list of dihedrals that will be optimized.

2. Dihedral step size :

   ```
   dihedral_step_size=<step size; default 15>
   ```

   Each dihedral identified for optimization is rotated through 360 degrees with a step-size of 15 degrees to generate QM target data. This step-size can be changed by adjusting the `dihedral_step_size` argument to increase or decrease the resolution of the dihedral scan.

3. Number of cores used for QM optimization:

   ```
   nproc=<number of core; default all available cores on␣
   ↪workstation/8 cores on HPC>
   ```

   At each scan point for each dihedral, a constrained QM optimization is performed. Psi4 can make use of multiple CPU cores to run this optimization more quickly. When the Optimizer is installed on a high-performance computing cluster (HPC), separate jobs are submitted for each of the identified dihedrals simultaneously, and each job requests `nproc` cores. When the Optimizer is installed on a workstation, then each scan point for each dihedral is run one after the other using all available cores on the workstation. This default behavior can be changed using the `nproc` argument.

4. Molecular-orbital theory:

   ```
   mo_theory=<MP2/HF; default MP2>
   ```

   Constrained QM optimization for each dihedral at each scan point is performed using Møller–Plesset perturbation theory (MP2, specifically the Psi4 DF-MP2/6-31G(d)

model chemistry). If the user wants to instead use Hartree-Fock (HF), then it can be set using `mo_theory`.

5. Energy difference cutoff to eliminate rotamers from QM optimization and subsequent fitting:

```
dg_cutoff=<energy_difference; default 100.>
```

The optimizer initially generates all the rotamers for each dihedral being optimized as required for the potential energy surface (PES) and performs a CGenFF energy optimization on each restrained rotamer. If the relative energy of each rotameter is greater than the `dg_cutoff` value (kcal/mo), that rotatamer is omitted from the QM optimization and subsequent parameter fitting. This is performed to eliminate rotamers with steric clashes that can be problematic for the QM optimization as well as lead to the parameter optimizer attempting to fit to high energy regions of the PES that are typically not sampled in MD simulations.

6. Specification of compute node for job submission:

```
hpc=<true/false; default true>
```

Installation of the CGenFF optimizer specifies the computer on which the jobs will be run, which is typically a local HPC cluster such that the default is `hpc=true`. However, if the user wants to run the job on their local workstation then `hpc=false` may be specified. Alternatively, if the default is false, `hpc=true` can be specified to run the job on the HPC cluster. This requires that the appropriate cluster be specified in the default installation.

7. Running jobs sequentially or in parallel when `hpc=true`.

```
jobtype=<seq/par; default seq>
```

When the QM jobs are submitted to an HPC queue they will run sequentially using the specified number of processors (`nproc`). However, if adequate compute resources are available it is possible to have all the QM jobs run simultaneously in parallel by specifying `jobtype=par` with each job using the specified number of processors (`nproc`). Care should be taken when using this option as the number of individual QM jobs can be quite large (n x 24 jobs where n is the number of dihedral parameters being optimized). However, use of the option can lead to the QM jobs finishing rapidly.

8. Setting the RMS energy difference to not redo the least-squares fitting using different multiplicities.

```
rmse_cutoff=<RMS energy cutoff to determine if additional␣
↪multiplicities should be tested; default=0.0>
```

The initial least-squares fit applies the multiplicities in the original CGenFF parameters assigned to the dihedral being optimized. From the fit the RMS difference between the target QM and optimized MM parameters is calculated. If that values is greater

than `rmse_cutoff` then least-squares fitting is performed with additional multiplicities. rmse_cutoff is set to zero by default to force the additional multiplicities to be included in the least-squares fitting. Note that the fitting may be repeated using the calculated QM data as described in the `Practical Considerations` section below.

11.5.2 Example Usecase

The following demonstrates usage of the CGenFF Parameter Optimizer. The input file `ethene_co_isoxazole.mol2` is available in `${SILCSBIODIR}/examples/cgenff/`. When running the Optimizer it is suggested that a subdirectory be created for each molecule and the job run from within that subdirectory.

```
${SILCSBIODIR}/cgenff-optimizer/optimize_cgenff_par mol=ethene_co_
↪isoxaole.mol2
```

The resulting output is a CHARMM-compatible CGenFF stream file named `ethene_co_isoxazole_optimized.str`. Since this molecule does not contain any dihedral parameter with penalty greater than 50, running the above command results in:

```
Nothing to optimize based on the penalty score threshold of  50
```

The user can look through the output stream file to identify dihedral parameters that may benefit from optimization:

```
DIHEDRALS
CG2DC3 CG2DC1 CG2O5  CG2R51     1.4000  2   180.00 ! NONAME.* , from␣
↪CG2DC3 CG2DC1 CG2O5 OG2D3, penalty= 26.5
HGA4   CG2DC1 CG2O5  CG2R51     0.0000  2   180.00 ! NONAME.* , from HGA4␣
↪CG2DC1 CG2O5 OG2D3, penalty= 26.5
CG2DC1 CG2O5  CG2R51 CG2R51     1.5850  2   180.00 ! NONAME.* , from␣
↪CG2O3 CG2O5 CG2R61 CG2R61, penalty= 46.5
CG2DC1 CG2O5  CG2R51 OG2R50     1.5850  2   180.00 ! NONAME.* , from␣
↪CG2O3 CG2O5 CG2R61 CG2R61, penalty= 49
OG2D3  CG2O5  CG2R51 CG2R51     1.5850  2   180.00 ! NONAME.* , from␣
↪OG2D3 CG2O5 CG2R61 CG2R61, penalty= 33.5
OG2D3  CG2O5  CG2R51 OG2R50     1.5850  2   180.00 ! NONAME.* , from␣
↪OG2D3 CG2O5 CG2R61 CG2R61, penalty= 13
CG2O5  CG2R51 CG2R51 CG2R52     0.0000  2   180.00 ! NONAME.* , from␣
↪CG2O1 CG2R51 CG2R51 CG2R52, penalty= 3
CG2O5  CG2R51 CG2R51 HGR51      1.0000  2   180.00 ! NONAME.* , from␣
↪CG2R51 CG2R51 CG2R51 HGR51, penalty= 16.5
CG2O5  CG2R51 OG2R50 NG2R50     8.5000  2   180.00 ! NONAME.* , from␣
↪CG2R51 CG2R51 OG2R50 NG2R50, penalty= 16.5
```

If the user now wants to fit CG2DC1 CG2O5 CG2R51 OG2R50, which has a penalty score of 49, the program can be re-run with a lowered penalty threshold of 48:

```
${SILCSBIODIR}/cgenff-optimizer/optimize_cgenff_par mol=ethene_co_
↪isoxazole.mol2 penalty=48
```

This will result in:

```
##############################################
for dihedral = CG2DC1 CG2O5 CG2R51 OG2R50, parameter penalty = 176.000000;
##############################################

Will be optimizing the parameters of the above listed dihedral(s)
```

Note that all parameters associated with dihedrals about a rotatable bond with a penalty score exceeding the penalty threshold will be optimized. Dihedrals that are considered rigid based on being located in rings, being double bonds and so on, are excluded from fitting. This selection process can lead to a situation in which two or more sets of dihedral parameters associated with the same rotatable bond are being fit independently. While each of those fits may appear to be successful, when those parameters are combined and applied to the molecules of interest, the conformational energy for rotation about that bond will likely be inaccurate. Such a situation should be avoided by setting the penalty tolerance to be higher that the penalty values for those parameters and one of the dihedrals selected for optimization as described in the following section.

If the user wants to fit the parameters for one or more specific dihedrals if none are selected based on the penalty score, or in addition to the automatically selected dihedrals, the user can manually supply the four atoms defined each specific dihedral on prompt:

```
${SILCSBIODIR}/cgenff=optimizer/optimize_cgenff_par mol=ethene_co_
↪isoxazole.mol2 penalty=50
```

will result in :

```
CHARMM General Force Field (CGenFF) program version 2.1.0
released the 27th of October 2016
Copyright (C) 2017 SilcBio LLC
and University of Maryland, School of Pharmacy. All Rights Reserved.

Now processing molecule sulf ...

##############################################
##############################################

Nothing to optimize based on the penalty score threshold of  50
```

(continues on next page)

(continued from previous page)

```
Do you want to perform optimization with anymore dihedrals? [Y/N; default␣
↪N]
```

To add to the list, type `Y`, and then supply dihedral(s) with the format `AtomType1 AtomType2 AtomType3 AtomType4`. The program then summarizes the updated-list:

```
List updated; will be attempting optimization with the following dihedrals

###############################################
CG2DC1 CG2O5 CG2R51 OG2R50
###############################################
```

The last column of the PDB file `<mol>_cgenff_atomtypes.pdb` created by the Optimizer identifies the atom type for each atom in the system which may be inspected to select specific parameters for optimization. For the current example, from the information in `ethene_co_isoxazole_cgenff_atomtypes.pdb`, atom 5 has the atom type `CG2R51`:

```
ATOM      1 C    NONA    1       0.759   0.680   1.188  1.00  0.00      ␣
↪CG2R52
ATOM      2 H    NONA    1       1.645   1.222   1.497  1.00  0.00       ␣
↪HGR52
ATOM      3 C    NONA    1      -0.353   1.220   0.530  1.00  0.00      ␣
↪CG2R51
ATOM      4 H    NONA    1      -0.508   2.244   0.227  1.00  0.00       ␣
↪HGR51
ATOM      5 C    NONA    1      -1.205   0.177   0.349  1.00  0.00      ␣
↪CG2R51
ATOM      6 C    NONA    1      -2.542   0.163  -0.296  1.00  0.00       ␣
↪CG2O5
ATOM      7 C    NONA    1      -3.305  -1.129  -0.378  1.00  0.00      ␣
↪CG2DC1
ATOM      8 H    NONA    1      -2.855  -2.041   0.043  1.00  0.00        ␣
↪HGA4
ATOM      9 C2   NONA    1      -4.530  -1.255  -0.944  1.00  0.00      ␣
↪CG2DC3
ATOM     10 H21  NONA    1      -5.050  -2.226  -0.980  1.00  0.00        ␣
↪HGA5
ATOM     11 H22  NONA    1      -5.045  -0.383  -1.387  1.00  0.00        ␣
↪HGA5
ATOM     12 O1   NONA    1      -3.016   1.204  -0.756  1.00  0.00       ␣
↪OG2D3
ATOM     13 O    NONA    1      -0.636  -0.949   0.875  1.00  0.00      ␣
↪OG2R50
```

(continues on next page)

(continued from previous page)

```
ATOM     14 N    NONA    1       0.613  -0.617   1.406  1.00  0.00      ␣
↪NG2R50
```

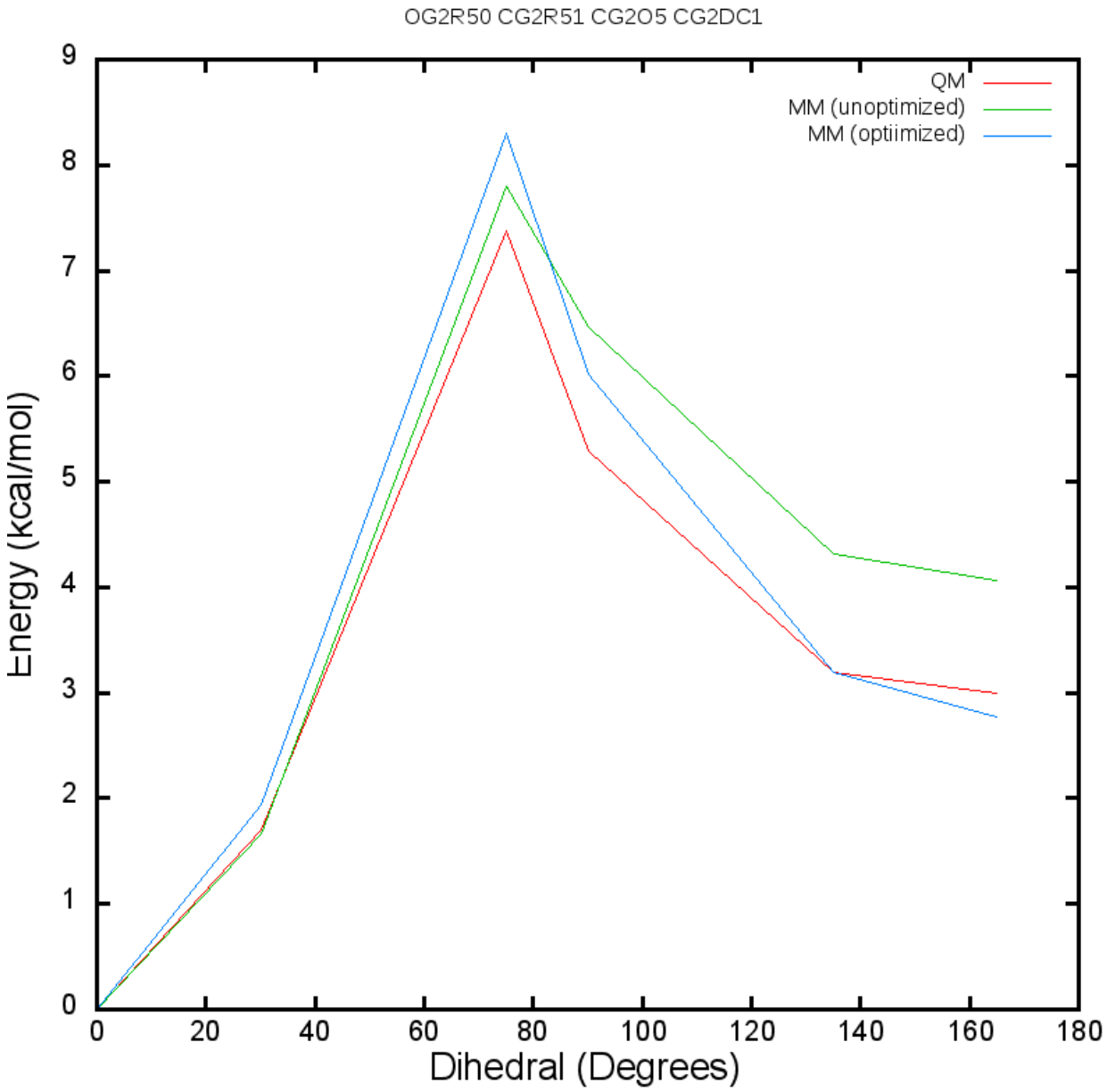
Following the identification of the dihedrals to the fitted, the Optimizer performs a complete molecular mechanics (MM) minimization of the molecule using the default CGenFF force field parameters. Next, each identified dihedral is independently rotated through 360 degrees with the specified step-size. Separate Psi4 jobs for constrained QM optimization around the specified dihedral are then performed either serially, one-by-one when running on a workstation, or in parallel when running on a HPC cluster.

The program waits on the QM jobs to finish. After the completion of the QM jobs, the MM energy is computed for each scan point. Both the QM and MM energies are then used by the lsfitpar program to fit the dihedral parameter. Updated parameters are output as `<mol>_optimized.str`.

Graphs of the dihedral scan energies are output in the files files `qm_mm_*.png`. If the computing system where you are running the Optimizer does not contain gnuplot or gnuplot linked against the png library, you may copy the `4plot` directory to a computing system that does and run the gnuplot files there.

Note that the most robust test of the optimized parameters involves testing them directly in a molecular modeling package with the new parameters added to the CGenFF parameter file (see below).

The following is a graph produced by the Optimizer for the `ethene_co_isoxazole` example and demonstrates the better agreement with the QM data following dihedral parameter optimization:



The default CGenFF parameter for dihedral `CG2DC1 CG2O5 CG2R51 OG2R50` was

```
CG2DC1 CG2O5  CG2R51 OG2R50     1.5850  2   180.00 ! RMSE = 0.468999
```

and the replacement parameter produced by the Optimizer is

```
CG2DC1 CG2O5  CG2R51 OG2R50     0.5728  2     0.00 ! RMSE = 0.111323
```

11.5.3 Performing Dihedral Optimization in the Background

The above example runs the optimization on the command line requiring the terminal in which the job is submitted to remain open. To run the optimization in the background, the 'nohup' command may be used as in the following example. Note that a small script would be required to identify specific dihedrals for optimization.

```
nohup $SILCSBIODIR/cgenff-optimizer/optimize_cgenff_par.sh mol=ethene_co_
↪isoxazole.mol2 > cgenff_opt_run1.txt &
```

11.5.4 Including New Dihedral Parameters with CGenFF

Currently, the optimized parameters generated by the CGenFF optimizer are NOT automatically added to the CGenFF parameter database. Accordingly, the user needs to manually add the parameters to the file using a standard editor (e.g., vi or emacs).

```
$SILCSBIODIR/data/par_all36_cgenff.prm
```

In the editor go the the DIHEDRALS section of the parameter file and then manually add the new parameters into the file. The new parameters are in filename_optimized.str found in the directory in which the optimization was run. "filename" corresponds to the molecule filehame.mol2 used to initiate the optimization. The optimized dihedral are at the end of the DIHEDRALS section of that file and are indicated by the comment containing the RMSE values obtained from the fitting of each dihedral parameter as follows.

```
CG2R51   CG2R51   NG311    CG331        3.8499  2   180.00 ! RMSE = 0.
↪58618
CG2R51   CG2R51   NG311    CG331        1.5145  3   180.00 ! RMSE = 0.
↪58618
CG3C52   CG2R51   NG311    CG331        2.1370  2   180.00 ! RMSE = 0.
↪245345
CG3C52   CG2R51   NG311    CG331        1.4058  3   180.00 ! RMSE = 0.
↪245345
```

Those parameters may be cut and paste directly into par_all36_cgenff.prm. Note that the file is generally protected and system administrator assistance is required to update this file. In the case of replacing dihedral parameters that are already in par_all36_cgenff.prm, the original parameters should be commented with a ! and the new parameters added.

11.5.5 Additional Output Information

Upon completion of the optimization the data generated is organized and placed in the subdirectory "raw_data" in which there is an additional subdirectory "psi4_calc" that contains files from the QM calculations.

- `optimize_cgenff_par.log`

  Summary of the dihedrals being optimized and the optimization process including the RMSE data and final parameters.

- `dih_params_to_optimize*.txt`

  Dihedral parameters initially selected and updated list targeted for optimization.

- `params*.{inp,out}`

  Inputs and outputs of MM minimizations to calculate the potential energy surfaces. Includes information on highly unfavorable conformations that were not included in the final parameter fitting.

- `params_aft_qm.out`

  Outputs of MM minimizations using the optimized parameters to calculate the potential energy surface.

- `lsqfit_op_0*.{inp,out}`

  Inputs and outputs for the parameter least-squares fitting using the original multiplicity along with fits using alternate multiplicities (`grep "Final RMSE" *.out` to see RMSE for all runs).

- `mm*.prm`

  Final parameters from the fits with the different multiplicities.

- `*.dat`

  Files containing dihedral angles and energies for fitting. Note that the `mm_init_ener_0.dat` is the MM PES with the targeted dihedral parameters set to zero.

11.5.6 Practical Considerations

The use of the CGenFF penalty scores to select dihedrals for optimization in conjunction with the selection of only rotatable bonds is designed for an automated approach to dihedral parameter optimization. However, this may often lead to multiple dihedrals being selected for optimization, which is computationally demanding, or no dihedrals being selected based on low or zero penalty scores, where zero indicates that the specific parameter is already in the CGenFF force field and, by default, such dihedrals will not be listed as available for optimization. However, in cases where the user is concerned about the accuracy of a particular rotatable bond even when it is already in the CGenFF parameter set (e.g., a rotatable bond involved in the link between two ring systems

that are part of a lead compound), then the user may specifically identify the atom types defining that dihedral parameter(s) and input them individually. While the penalty scores represent a useful metric by which to judge the quality of a dihedral parameter it should be noted that the CGenFF penalty scores are based on analogy to known parameters. Accordingly, a parameter with a high penalty may yield acceptable conformational energies, while a parameter already in CGenFF (i.e., penalty = 0) may not yield an acceptable energy surface due to, for example, the terminal rings about the associated bond creating a significantly different context than that in which the dihedral parameter was initially optimized.

An important consideration is molecular size. As the fitting procedure requires QM calculations the size of the molecule under study significantly impacts the speed of the overall fitting procedure. To limit the computational demand while achieving the desired accuracy in the parameters it is suggested that compounds be subdivided into model compounds extracted from that full compound that contain two terminal ring systems along with the linker between those rings. The number of substituents on each ring should be limited to those deemed essential to represent the chemical character of the rings while minimizing the number of substituents to limit computational costs. For example a compound with three ring systems with two linkers would be subdivied into two model compounds with two rings each, with one of the rings common to both model compounds. Once the two individual CGenFF-optimizer runs are finished, the new parameters may be combined and added to your the CGenFF parameter files.

During parameter fitting, the program initially uses the dihedral parameter multiplicities assigned by the CGenFF program. Once the RMSE is returned, the program then does additional fitting with a multiplicity of 1, then multiplicities of 1 and 2, 1, 2 and 3 and 1, 2, 3 and 6. The current implementation outputs the results from all the multiplicities tested. As the lowest RMSE will generally be with multiplicities 1,2,3,6 the user may select a set of parameters with different multiplicities. Those parameters are in the files `raw_data/mm*.prm`. Note that the multiplicities of 4 and 5 are omitted as these are rarely used for rotatable bonds even though, in same cases, the CGenFF program will assign dihedral parameters with these multiplicities and the program will initially include a dihedral with a multiplicity of 4 or 5 in the initial fitting.

If the level of agreement using the final parameters is not satisfactory it suggests hysteresis in the energy surface. To overcome this the user may consider focusing the fitting on an 'appropriate' part of the energy surface by trimming down some data-points on the QM & MM energies to get a better fit to the low energy region of the energy surface. To achieve this consider running `${SILCSBIODIR}/cgenff-optimizer/refit_par_with_subsurface dih_qm_<dihno>.dat dih_mm_<dihno>.dat` (Under development).

Upon completion of the fitting a subdirectory 'raw_data' is created that contains an extensive number of data file with the various QM energies, MM energies and additional information. In addition, the subdirectory 'raw_data/psi4_calc' contains the inputs and outputs from the psi4 QM optimizations. These files may be used for additional fitting or analysis.

In cases where the least-squares fitting is not completed successfully, possibly due to all the QM jobs not finishing, fitting alone may be performed if all the QM data is directly available in subdirectory `psi4_calc` (not `raw_data/psi4_calc`). For example, if a subset of the QM jobs did not finished the job input scripts in `psi4_calc` (e.g., `psi4_inp_xxxx_1_23.inp`) maybe submitted directly

to complete those jobs followed by resubmission of the cgenff-optimizer command from the parent directory.

CHAPTER

TWELVE

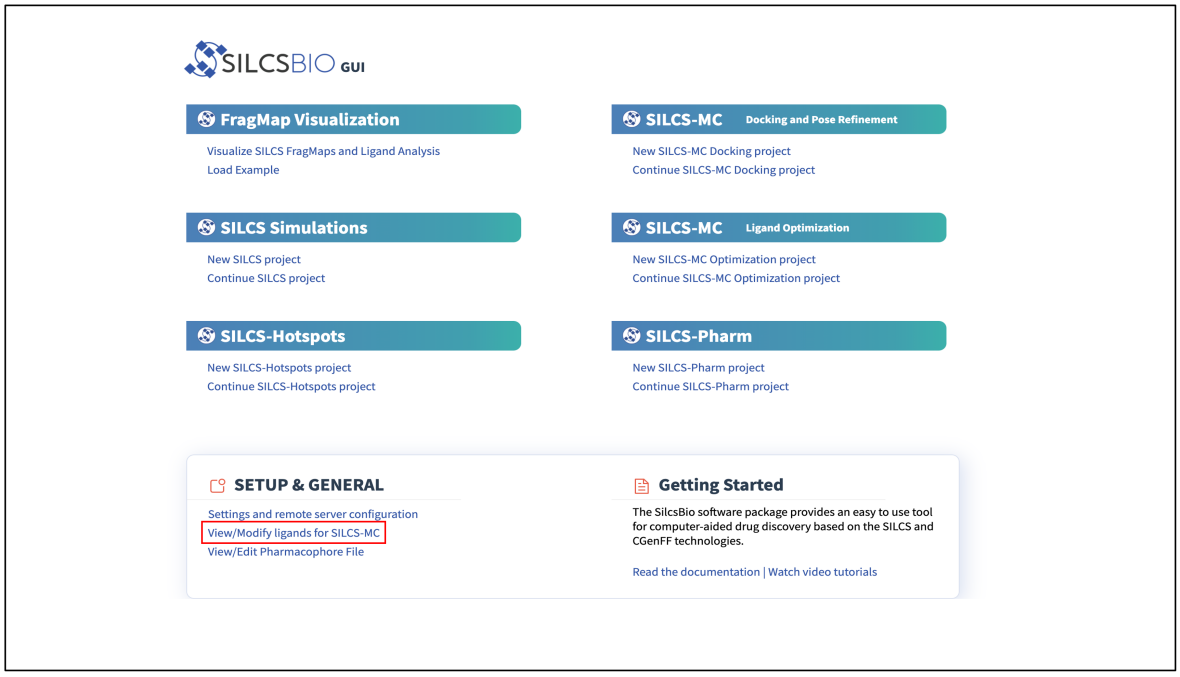
UTILITIES

12.1 Chemical Group Transformations

SILCS-MC ligand optimization and SSFEP both entail computations on derivatives of a parent ligand. Additionally, the user may wish to perform SILCS docking using a series of structures chemically related to a reference compound (parent ligand). Therefore, a task common to these use cases is that of performing chemical group transformations on a parent ligand to create a series of chemically-related compounds.

12.1.1 Ligand Modifications Using the SilcsBio GUI

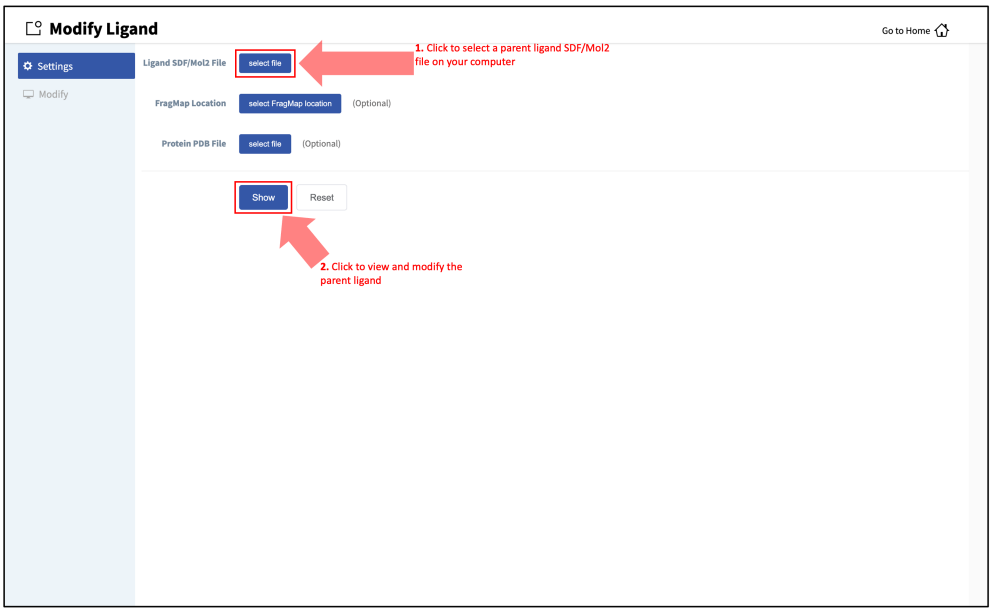
The SilcsBio GUI provides an intuitive way to modify ligands directly in the context of SILCS-MC ligand optimization (see *SILCS-MC Ligand Optimization Using the SilcsBio GUI*) and SSFEP (see *SSFEP Suite Quickstart*). You can also use the SilcsBio GUI to perform chemical transformations independently of any other task by choosing *Modify ligand for SILCS-MC* from the Home page of the Small Molecule Suite. The utility is also available from the Home page of the SSFEP and CGenFF Suites through *Modify ligand for SSFEP* and *Modify ligand for CGenFF*



Output Mol2 files will be saved to a directory of your choosing on the computer where you are running the SilcsBio GUI. Instructions for building ligand modifications through *Modify ligand for SILCS-MC/SSFEP/CGenFF* from the Home page are provided below.

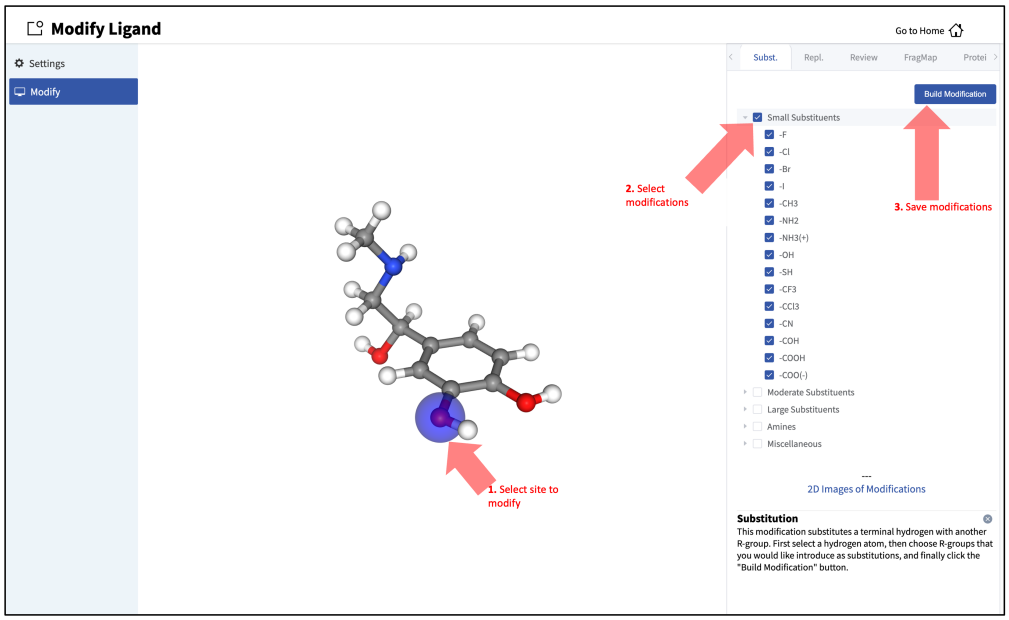
1. **Enter the parent ligand SDF or Mol2 file:**

   Provide the SDF or Mol2 file of the parent ligand structure. To guide the selection of sites to modify and ligand modifications to apply, the user may also provide the structure of the target protein and the corresponding FragMaps. If the target protein and FragMaps are loaded, ensure that the parent ligand is appropriately positioned within the binding pocket. Click "Show" to proceed.



2. **Add ligand modifications:**

   The GUI will display the parent ligand. First, select the atom to be modified. Then, select the desired modifications from the "Subst." (Substitution) or "Repl." (Replacement) tab in the right-hand panel. Substitution is used to substitute an atom with a functional group. Replacement is used to replace an atom in a ring or aliphatic acyclic group with another functional group that preserves the connectivity and valence of the ring or chain. The list of modification types in the GUI covers a very broad range of chemical functionality and size (see *Available Ligand Modifications in the SilcsBio GUI* for the list of possible modifications in SilcsBio GUI). Press the "Build Modification" button in the panel to build the selected modifications at the selected modification site.

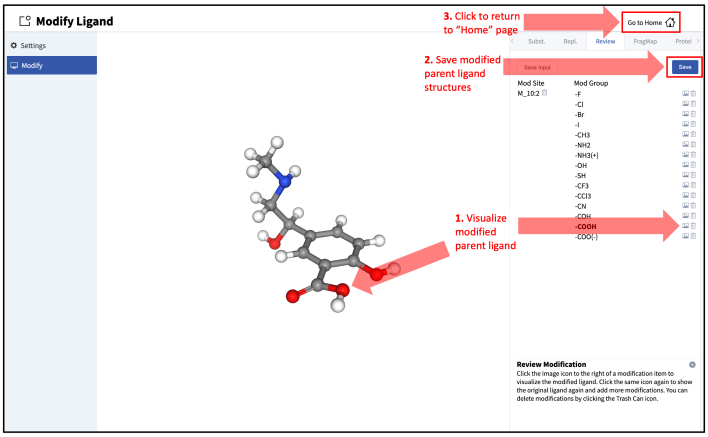


3. **Save modified ligands:**

   Use the "Review" tab to confirm and save the desired modifications.

   Valid modifications will have a small Image icon as well as a small Trash Can icon. Clicking on the Image icon will show the modification in the center panel. Clicking on it again will show the parent ligand. Clicking on the Trash Can icon will delete the proposed modification from the "Review" list. Additional modifications can be included in the list by re-entering the "Subst." and "Repl." tabs. Once the list of modifications is complete, save the modifications by clicking the "Save" button under the "Review" tab. Output Mol2 files will be saved to a user-specified directory of your choice on the local machine where the SilcsBio GUI is being run.

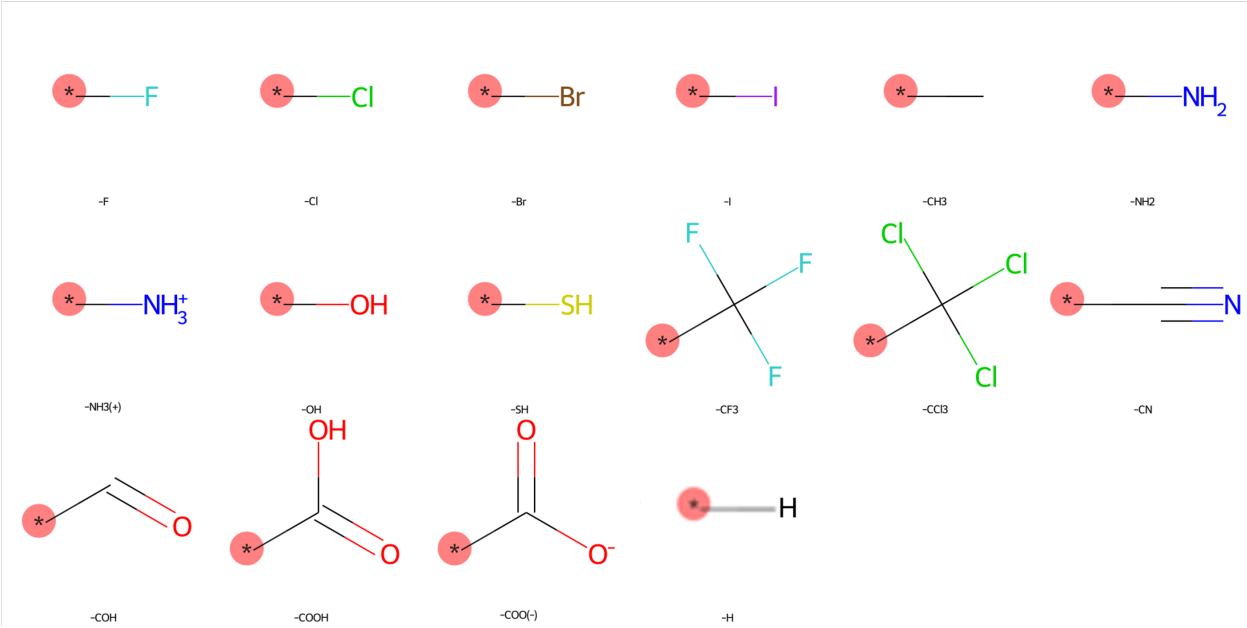
   After saving the modified ligands, click "Go to Home" at the top right corner of the page to return to the "Home" page.



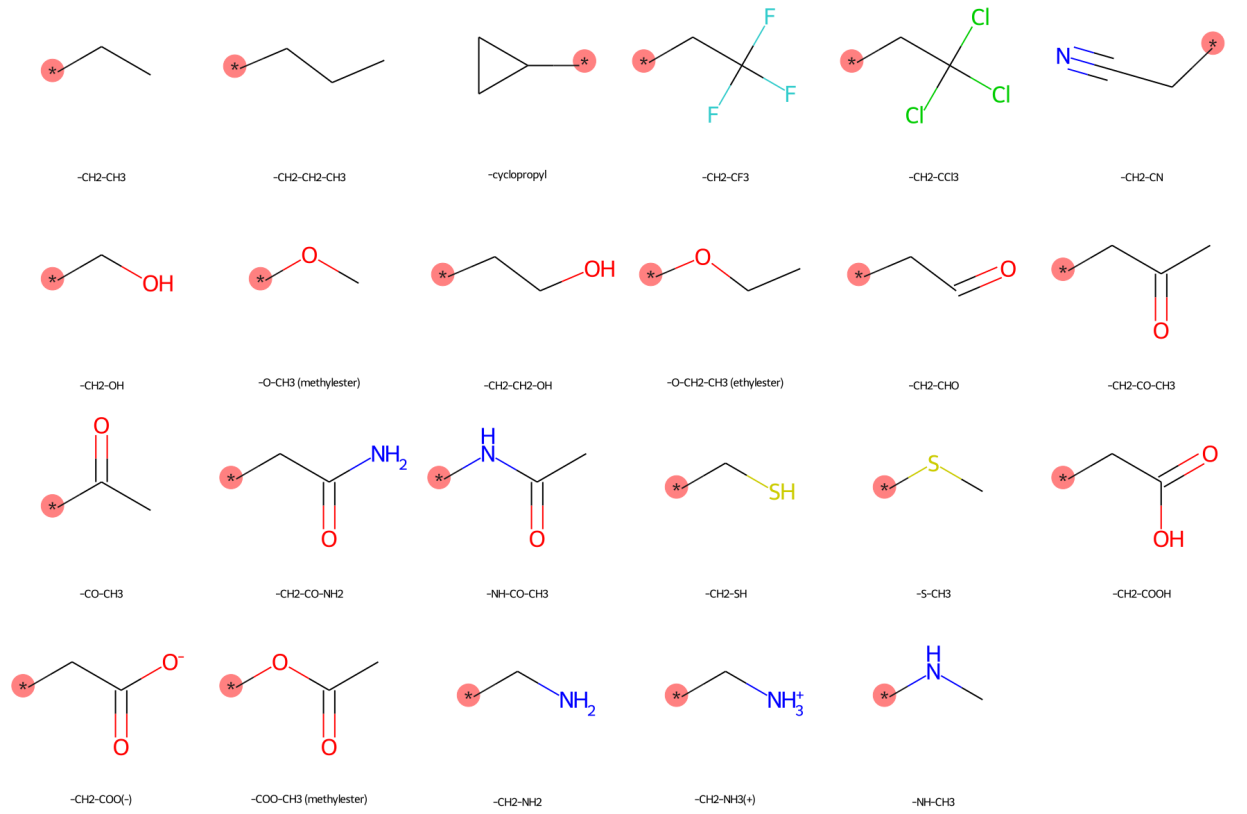
Available Ligand Modifications in the SilcsBio GUI

The SilcsBio GUI provides a wide array of ligand modifications. The readily available ligand substitution modifications are listed below. In the images below, the red circle and asterik indicate the location at which the substitution will be added to the input ligand. In addition to these substitutions, please see the "Repl." tab within the GUI itself for readily available replacement options.

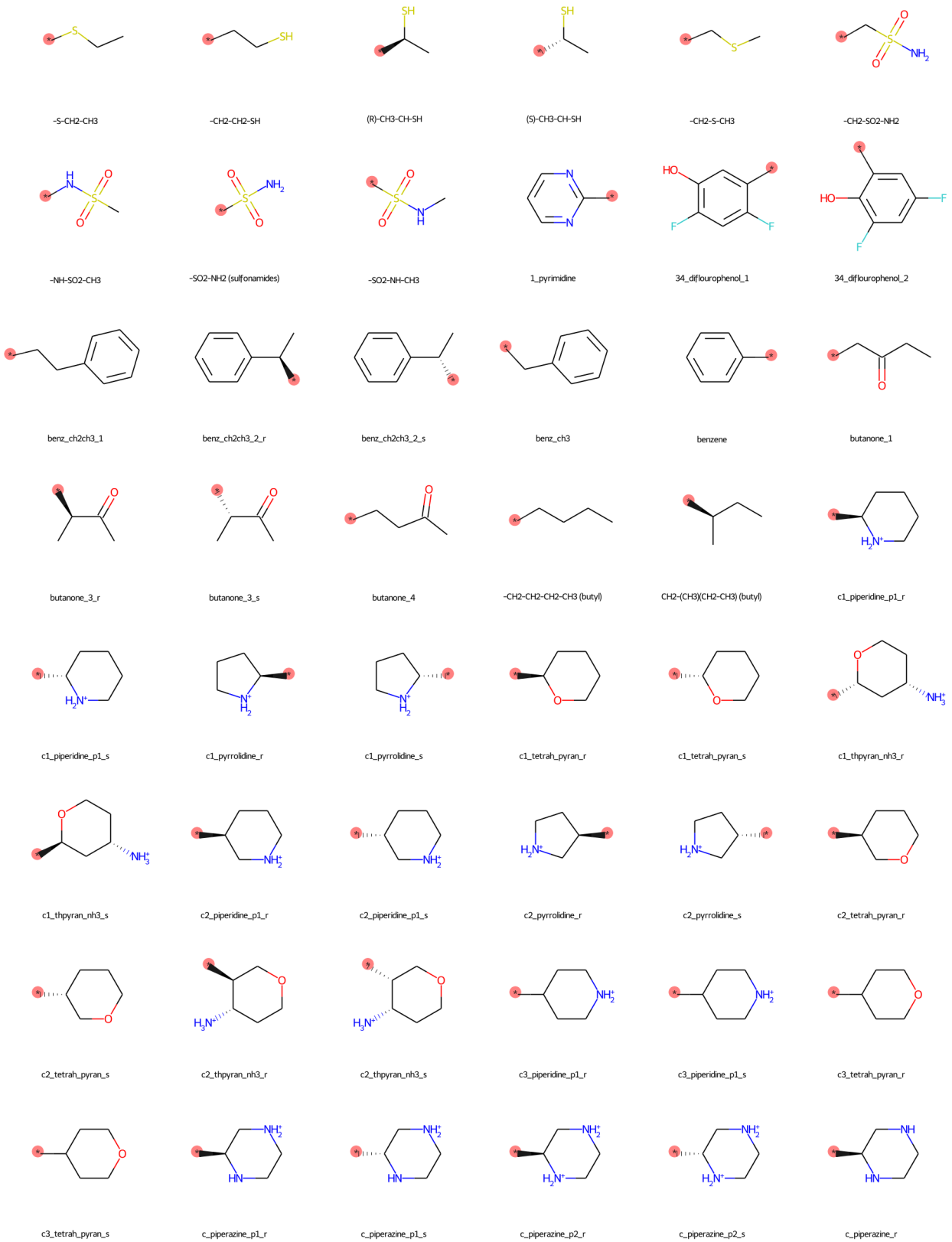
Small Substituents

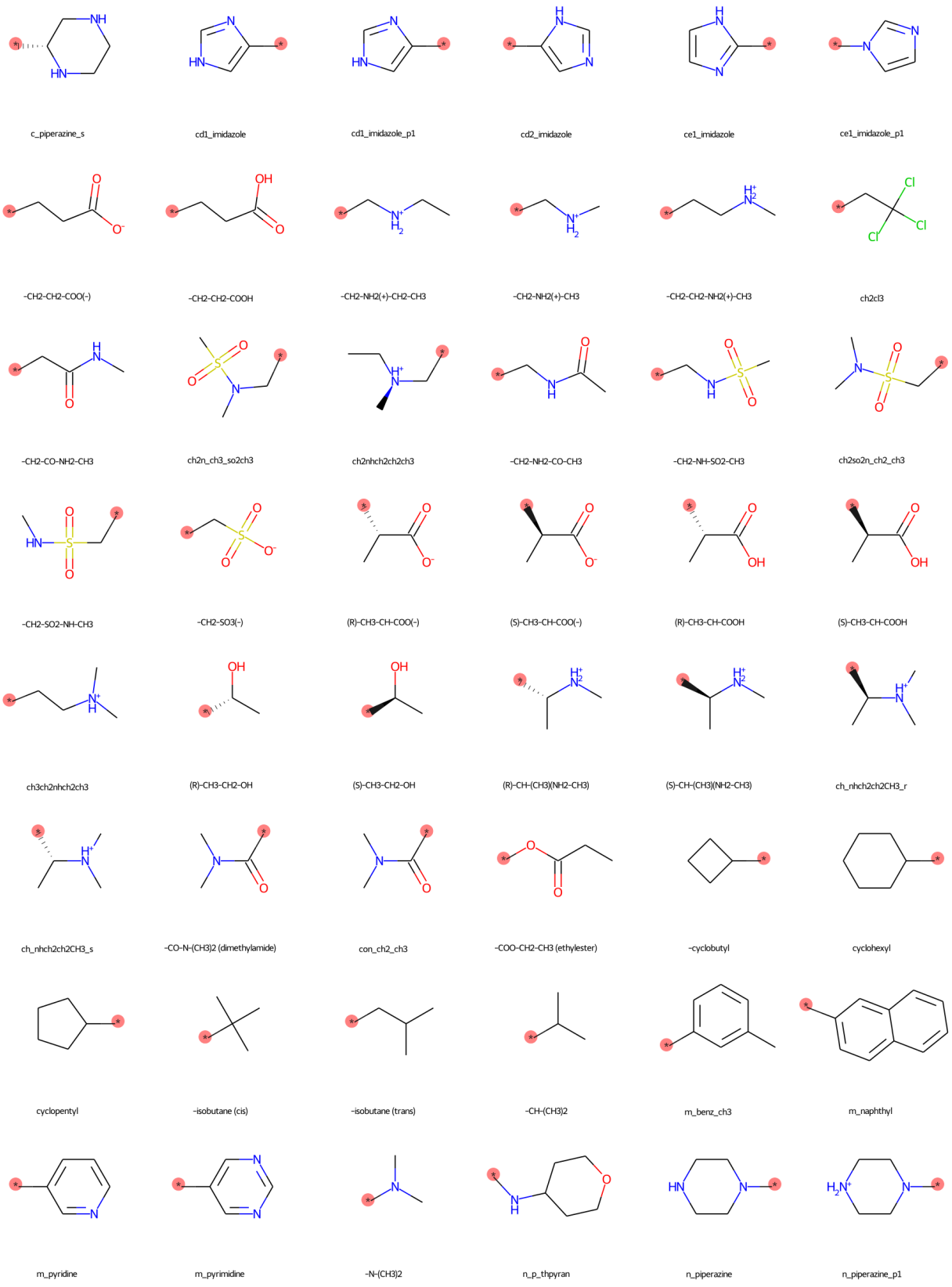


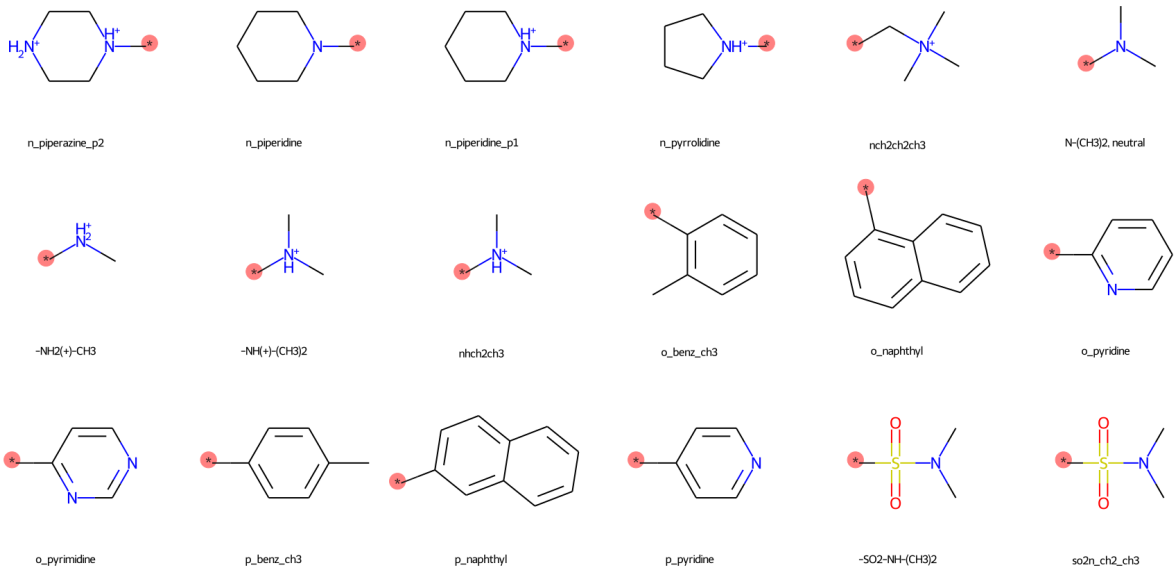
Moderate Substituents



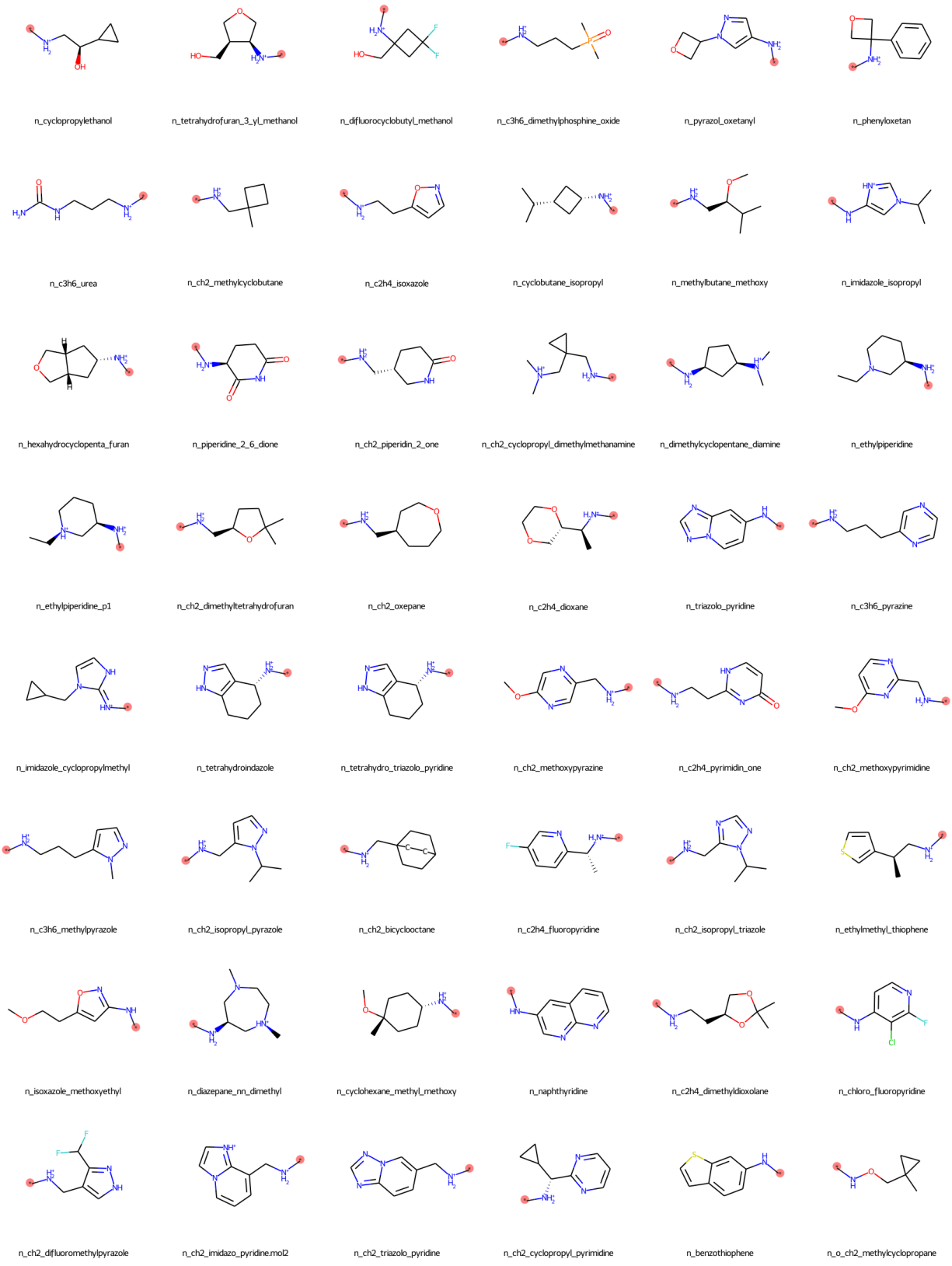
Large Substituents

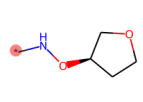



c_piperazine_s
cd1_imidazole
cd1_imidazole_p1
cd2_imidazole
ce1_imidazole
ce1_imidazole_p1
-CH2-CH2-COO(-)
-CH2-CH2-COOH
-CH2-NH2(+)-CH2-CH3
-CH2-NH2(+)-CH3
-CH2-CH2-NH2(+)-CH3
ch2cl3
-CH2-CO-NH2-CH3
ch2n_ch3_so2ch3
ch2nhch2ch2ch3
-CH2-NH2-CO-CH3
-CH2-NH-SO2-CH3
ch2so2n_ch2_ch3
-CH2-SO2-NH-CH3
-CH2-SO3(-)
(R)-CH3-CH-COO(-)
(S)-CH3-CH-COO(-)
(R)-CH3-CH-COOH
(S)-CH3-CH-COOH
ch3ch2nhch2ch3
(R)-CH3-CH2-OH
(S)-CH3-CH2-OH
(R)-CH-(CH3)(NH2-CH3)
(S)-CH-(CH3)(NH2-CH3)
ch_nhch2ch2CH3_r
ch_nhch2ch2CH3_s
-CO-N-(CH3)2 (dimethylamide)
con_ch2_ch3
-COO-CH2-CH3 (ethylester)
-cyclobutyl
cyclohexyl
cyclopentyl
-isobutane (cis)
-isobutane (trans)
-CH-(CH3)2
m_benz_ch3
m_naphthyl
m_pyridine
m_pyrimidine
-N-(CH3)2
n_p_thpyran
n_piperazine
n_piperazine_p1

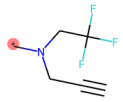

n_piperazine_p2
n_piperidine
n_piperidine_p1
n_pyrrolidine
nch2ch2ch3
N-(CH3)2, neutral
-NH2(+)-CH3
-NH(+)-(CH3)2
nhch2ch3
o_benz_ch3
o_naphthyl
o_pyridine
o_pyrimidine
p_benz_ch3
p_naphthyl
p_pyridine
-SO2-NH-(CH3)2
so2n_ch2_ch3

Amine Substituents

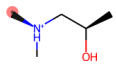




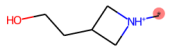
n_o_tetrahydrofuran



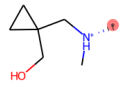
n_trifluoroethyl_propyne



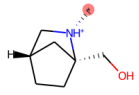
n_methylamino_propan_2_ol



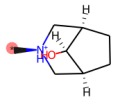
n_azetidin_ethanol



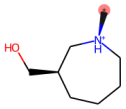
n_methylamino_methyl_cyclopropyl



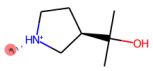
n_azabicyclo_heptanyl_methanol



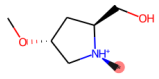
n_azabicyclo_octanol



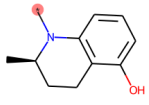
n_azepanylmethanol



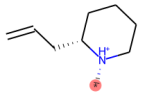
n_pyrrolidinyl_propanol



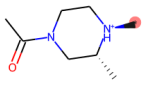
n_methoxypyrrolidine_methanol



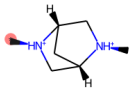
n_methyl_tetrahydroquinolinol



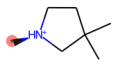
n_allylpiperidine



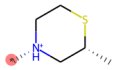
n_methylpiperazine_ethan_one



n_methyl_diazabicycloheptane

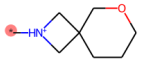


n_dimethylpyrrolidine

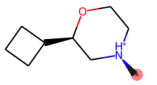


n_methylthiomorpholine

n_methyl_diazaspiro_octane

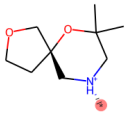


n_oxa_azaspiro_nonane

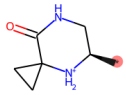


n_cyclobutylmorpholine

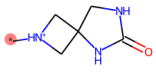
n_dimethylthiomorpholine_dioxide



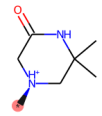
n_dimethyl_dioxa_azaspiro



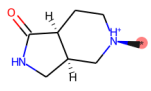
n_diazaspiro_octan_one



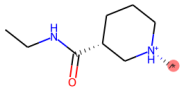
n_triazaspiro_octan_one



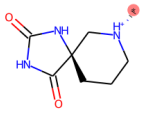
n_dimethylpiperazin_one



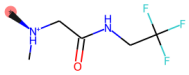
n_octahydro_pyrrolo_pyridin_one



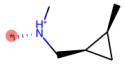
n_ethylpiperidine_carboxamide



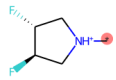
n_triazaspiro_decane_dione



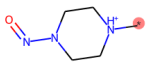
n_methylamino_trifluoroethyl_acetamide



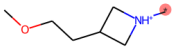
n_methyl_methylcyclopropylmethanamine



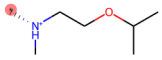
n_difluoropyrrolidine



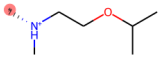
n_nitrosopiperazine



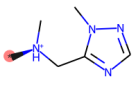
n_methoxyethyl_azetidine



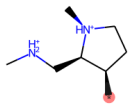
n_isopropoxy_methylethan_amine



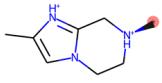
n_methyl_methyl_triazol_methanamine



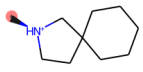
n_methyl_methylpyrrolidin_methanamine



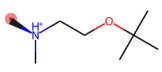
n_furan_pyrrolidine



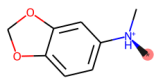
n_methyl_tetrahydroimidazo_pyrazine



n_azaspiro_decane



n_tert_butoxy_methylethan_amine



n_methylbenzo_dioxol_amine

n_methyl_pyrrolidin_pyrazole

n_cyclopropyloxazol_methylmethanamine

n_tolyl_pyrrolidine

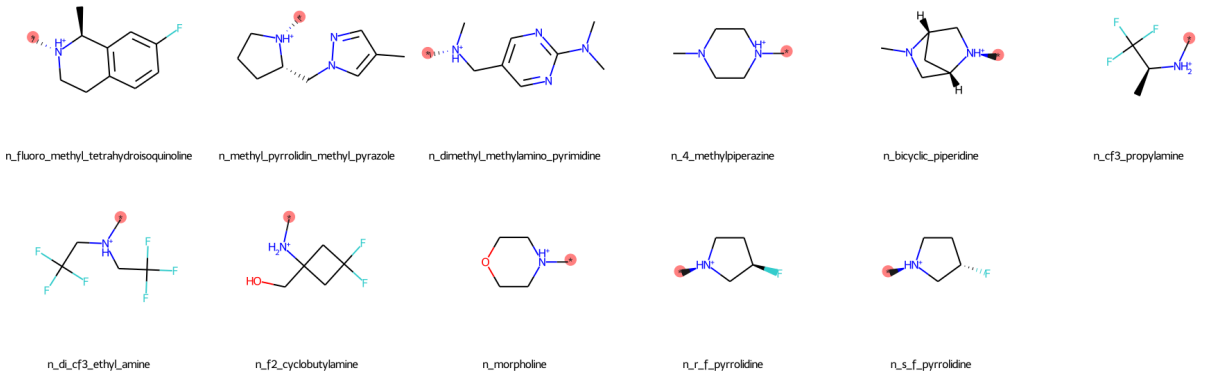
n_benzo_triazol_methylmethanamine

n_indoline_carboxamide

n_piperidin_pyridine

n_pyridin_ethyl_cyclopropanamine

n_pyrrolidin_oxy_pyridine



12.1.2 Ligand Modifications Using the CLI

To perform chemical group transformations using the CLI, you will need to create a text file `modifications.txt` with instructions listing the desired modifications to your parent ligand.

Three types of modifications can be performed, as listed in the table.

Command	Modification
JOIN	Join a fragment (mol2) to the parent at a defined site
REPL	Replace an atom at a defined site on the parent with a fragment (mol2)
MUDE	Change an atom at a defined site on the parent to another atom-type

Table 12.1: Available modification types.

An example ligand modification input file is povided with your SilcsBio server software, `${SILCSBIODIR}/examples/ssfep/modification.txt`.

For details on how to modify and process this file to create Mol2 files containing modifications to your parent ligand, run the following command:

```
${SILCSBIODIR}/programs/chemmod -h
```

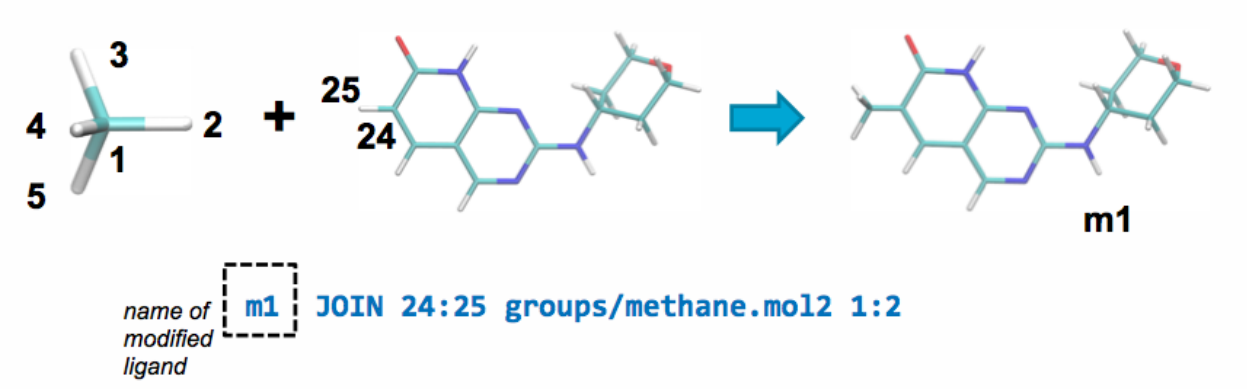
Your SilcsBio server software comes with a large number of fragments in Mol2 format that can be added/joined to your parent ligand. These fragments are found in:

```
${SILCSBIODIR}/data/groups
```

Modification Examples

JOIN Example

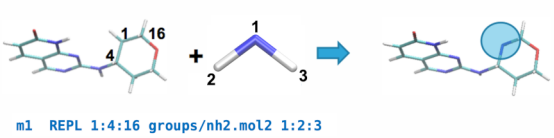
A **JOIN** operation can be performed between the parent and a `ch4.mol2` fragment by adding the following line to `modifications.txt`:



This line will join atom 24 in the parent ligand with atom 1 in `methane.mol2` and delete atoms 25 and 2 in the parent ligand and the joined fragment, respectively.

REPL Example

A **REPL** operation can be performed between the parent and an `nh2.mol2` fragment by adding the following line to `modifications.txt`:



This line will replace atom 1 of the parent ligand with atom 1 of the fragment by aligning atoms 2 & 3 of the fragment with atoms 4 & 16 of the parent, respectively, and replacing the carbon in the ring with a nitrogen atom.

MUDE Example

The same transformation in the previous section can also be achieved using a **MUDE** operation:

```
m2  MUDE MU 1:7 DE 21
```

This line will mutate atom 1 (atom index number) in the parent with nitrogen (atomic number 7) along with deleting the hydrogen (atom index number 21) attached to the parent carbon.

Ligand Decoration

To evaluate multiple modifications to a single site on the parent ligand, use the following syntax:

```
m1  JOIN 24:25 <filename>
```

Replace <filename> with the name of a text file containing each of the multiple modifications, with one modifications per line. Examples of such text files are `list.txt` and `small_join_list.txt` located in the `${SILCSBIODIR}/data/` folder, and you can use these example files as the basis for your own custom file that includes modifications using Mol2 files from `${SILCSBIODIR}/data/groups` or your own custom Mol2 files. Be careful to pay attention to chemistry; if a modification in your text file is not suitable for a site, you can comment it out using `!` at the beginning of that line. Based on the above `JOIN` line, Mol2 output files from running `${SILCSBIODIR}/programs/chemmod` will all begin with the prefix `m1_`.

12.2 Atom Selection Syntax

12.2.1 Atom Selection in the SilcsBio GUI

The SilcsBio GUI uses the same atom selection syntax as the NGL molecular structure viewer. This syntax is described at https://nglviewer.org/ngl/api/manual/selection-language.html and is reproduced below for convenience.

Keywords

- all, *
- sidechain
- sidechainAttached (not backbone or .CA or (PRO and .N))
- backbone
- protein
- nucleic
- rna
- dna
- hetero
- ligand ((not polymer or hetero) and not (water or ion))
- ion

- saccharide/sugar
- polymer
- water
- hydrogen
- helix
- sheet
- turn (not helix and not sheet)
- small (Gly or Ala or Ser)
- nucleophilic (Ser or Thr or Cys)
- hydrophobic (Ala or Val or Leu or Ile or Met or Pro or Phe or Trp)
- aromatic (Phe or Tyr or Trp or His)
- amid (Asn or Gln)
- acidic (Asp or Glu)
- basic (His or Lys or Arg)
- charged (Asp or Glu or His or Lys or Arg)
- polar (Asp or Cys or Gly or Glu or His or Lys or Arg or Asn or Gln or Ser or Thr or Tyr)
- nonpolar (Ala or Ile or Leu or Met or Phe or Pro or Val or Trp)
- cyclic (His or Phe or Pro or Trp or Tyr)
- aliphatic (Ala or Gly or Ile or Leu or Val)
- bonded (all atoms with at least one bond)
- ring (all atoms within rings)

Expressions

- residue number: 1, 2, 100
- residue number range: 3-40 (Note that around the dash - no spaces are allowed)
- chain name: :A
- atom name: .CA, .C, .N, …
- model: /0, /1, …
- residue name: ALA, GLU, SOL, DMPC, …
- numeric residue name: [032], [1AB], …

- list of residue names: [ALA,GLU,MET], [ARG,LYS], ...
- element name: _H, _C, _O, ...
- alternate location: %A, %B, ... or % for non-alternate location atoms
- insertion code: ^A, ^B, ... or ^ for residues with no insertion code

Some of these expressions can be combined (in this order) - residue number (range), insertion code, chain name, atom name, alternate location, model - like this

```
// select C-alpha atoms of residue 10 with insertion code A
// from chain F in model 0 at alternate location C
10^A:F.CA%C/0
```

which is the same as

```
10 and ^A and :F and .CA and %C and /0
```

Single expressions may be left out as long as the order (see above) is kept, for example:

```
:A/0 # select chain A from model 0
```

Atomindex

A list of atom indices can be given as a comma seperated list (no spaces in between) prefixed with the @ character.

```
@0,1,4,5,11,23,42
```

Logical Operators (in Order of Binding Strength)

- NOT
- AND
- OR

Additionally, parentheses () can be used for grouping:

```
:A and ( 1 or 10 or 100 ) # select residues 1, 10 and 100 from␣
↪chain A
```

12.2.2 Atom Selection in the SilcsBio CLI

The SilcsBio CLI tools that allow users to select specific atoms follow the same atom selection syntax as GROMACS in the command line. This syntax is described in the help menu when entering the command `echo h | $GMXDIR/gmx make_ndx -f <structure file>`. The syntax is also described here for convenience.

Keywords and Operators

Table 12.2: Table of Operators

Operators	Description	Usage
Identifiers		
`nr`	Selects an index group by number or quoted string.	`1`
`'a' nr`	Selects atoms by index. Atom numbering starts at 1.	`'a' 1 2 3 10`
`'a' name`	Selects atoms by atom name.	`'a' CA`
`'t' type`	Selects atoms by atom type. A run input file is required.	`'t' CA`
`'r' nr`	Selects residues by number.	`'r' 31 44`
`'r' name`	Selects residues by resname.	`'r' ASP VAL TRP`
`'ri' nr`	Selects residues by number.	`'r' 31 44`
`'chain' name`	Selects atoms by chain identifier(s). Not available for use with `.gro` file as input.	`'chain` A B C`
Modifiers		
`-`	Selects indicies within the specified range.	`'a' 1-4 'r' 5-10`
`?`	Wildcard that matches any one character	`'a' name1? name2?`
`*`	Wildcard allowed at the end of a name to match a pattern of any length	`'r' name1* name2*`
Logical Operators		
`'!'`	Not.	`'! r 1'`
`'&'`	And.	`r 1 & 1`
`'|'`	Or.	`'r 1 | r 29'`

continues on next page

Table 12.2 – continued from previous page

Operators	Description	Usage
Commands		
`'name'`	Rename group `nr` to `'name'`.	`'name' nr name`
`'del'`	Deletes specified group(s).	`'del' 1 - 3`
`'keep'`	Deletes all groups except for the specified group.	`'keep' 1`
`'case'`	Make all name comparisons case (in)sensitive.	
`'splitch'`	Split specified group into chains using CA distances.	`'splitch' 1`
`'splitres'`	Split specified group into residues.	`'splitres' 1`
`'splitat'`	Split specified group into atoms.	`'splitat'' 1`
`'res'`	Interpret numbers in specified group as residue numbers.	`'res' 1`
`Enter`	List currently defined groups and commands.	
`'l'`	List the residues.	
`'h'`	Show the help menu.	
`'q'`	Save and quit	

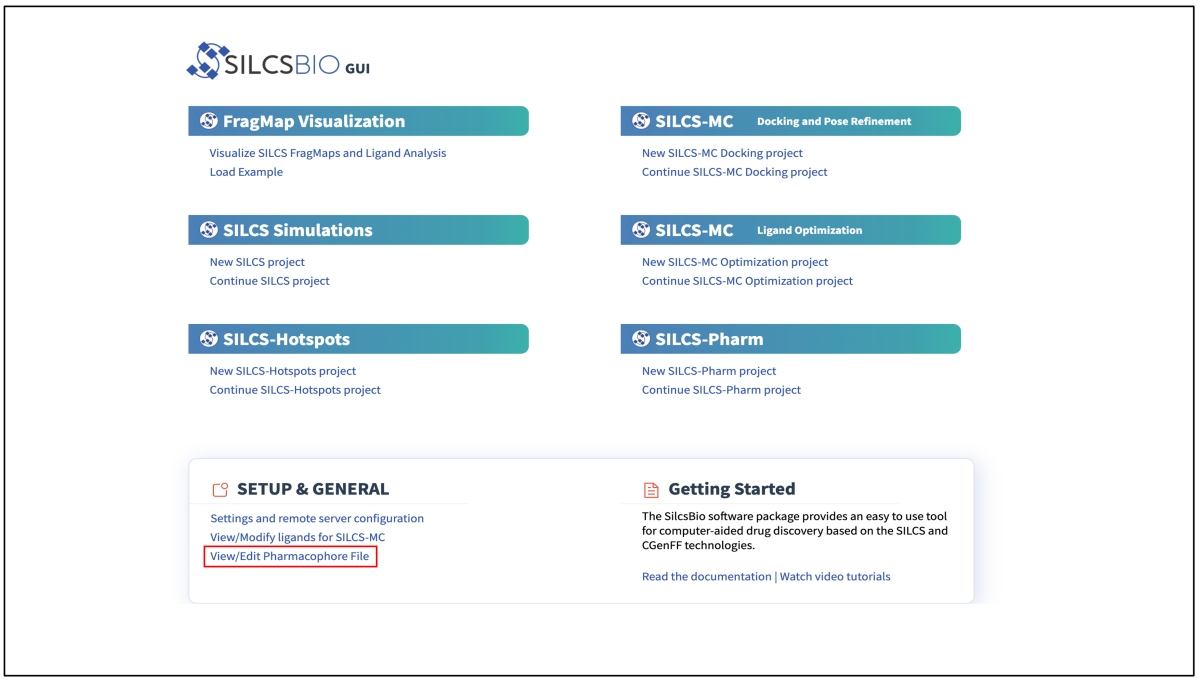
Examples

Table 12.3: Table of Examples

Selection	Description
`2 | 4 & r 3-5`	selects all atoms from group 2 and 4 that have residue numbers 3, 4 or 5
`a C* & !a C CA`	selects all atoms starting with 'C' but not the atoms 'C' and 'CA'
`"protein" & ! "backb"`	selects all atoms that are in group 'protein' and not in group 'backbone'

12.3 View and Edit Pharmacophore Files

The SilcsBio GUI provides a convenient platform to view and edit existing pharmacophore files. The pharmacophore model viewing and editing platform can be accessed by choosing *View/Edit Pharmacophore File* from the Home page of the Small Molecule Suite.

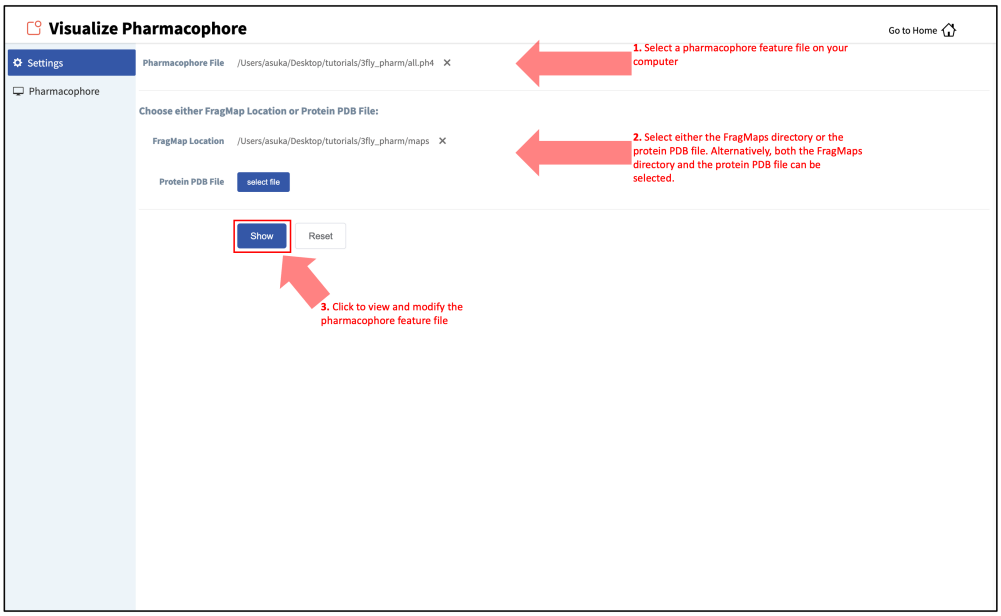


Using this tool, users can view the pharmacophore feature in the context of the target protein and the corresponding FragMaps, measure distances between pharmacophore features, select or deselect pharamacophore features in the pharmacophore model, adjust the feature radii, and save the modified pharmacophore model into a new pharmacophore file.

Instructions for viewing and editing exisitng pharmacophore files are provided below.

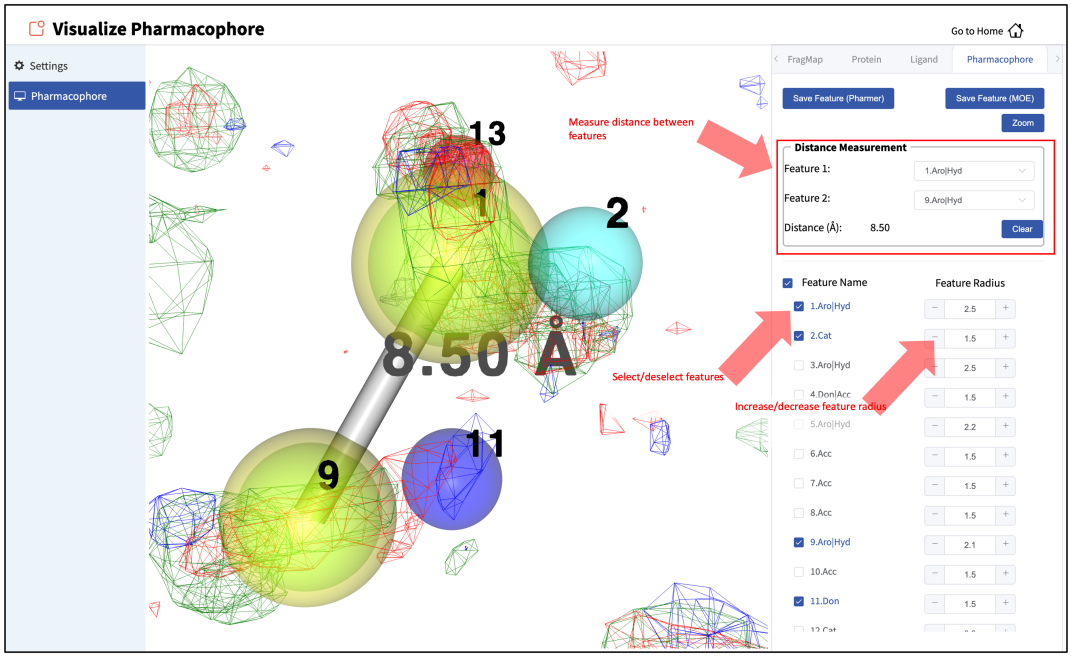
1. **Enter the pharmacophore file:**

   Provide the pharmacophore file to be viewed or modified. To guide the selection of pharmacophore features to modify, the user may also provide the structure of the target protein and the corresponding FragMaps. If the target protein and FragMaps are loaded, ensure that the pharmacophore file was created using the same input protein and FragMaps. Click "Show" to proceed.



2. **Edit the pharmacophore feature selection:**

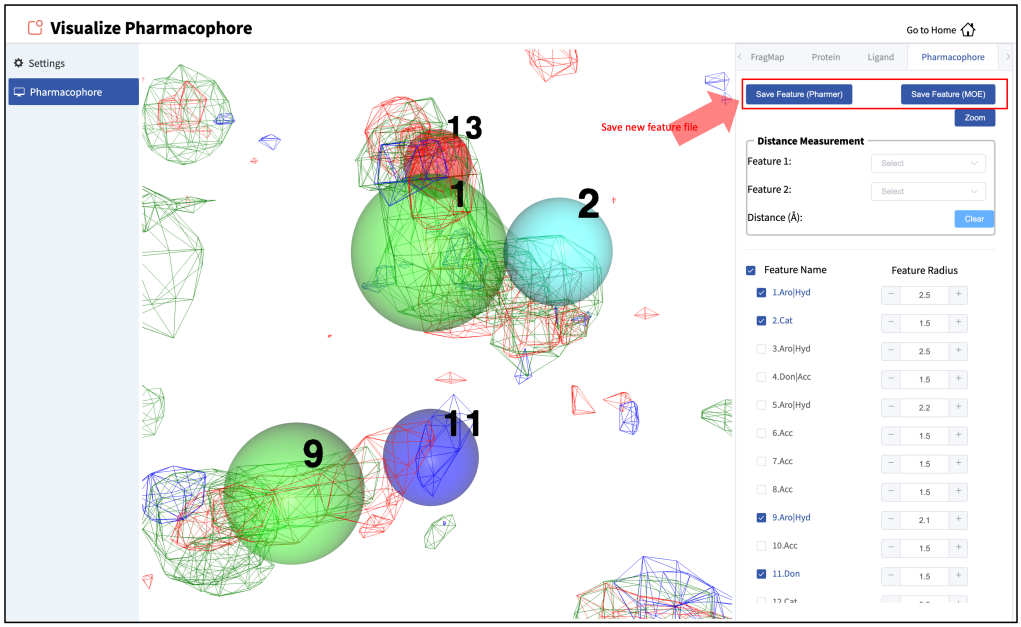
   After clicking "Show", the GUI will display the pharmacophore features. The "Pharmacophore" tab will list all the pharmacophore features from the input pharmacophore file. From this window, the user can measure the distance between features, select or deselect features, and increase or decrease feature radii.



3. **Save the new pharmacophore model:**

   After finalizing pharmacophore feature selection and adjustment, the new pharmacophore model can be saved by clicking either of the "Save Feature" buttons. The pharmacophore

model can be saved in `.ph4` format compatible with either Pharmer or MOE.



Note: The exclusion pharmacophore features are NOT supported by Pharmer.

12.4 Rename FragMaps Directory

Users may wish to rename their SILCS FragMaps directory. This can be achieved conveniently using the following command:

```
$SILCSBIODIR/utils/rename_fragmaps_directory.sh <current_fragmapsdir>
↪<new_fragmapsdir> \
                                                <current_protname> <new_
↪protname>
```

Required parameters:

- Path to original SILCS FragMaps directory (`<current_fragmapsdir>`)
- Path to renamed SILCS FragMaps directory (`<new_fragmapsdir>`)

Optional parameters:

- Name of original protein PDB file (`<current_protname>`)
- Name of renamed protein PDB file (`<new_protname>`)

Examples:

```
$SILCSBIODIR/utils/rename_fragmaps_directory.sh silcs_fragmaps_1abc silcs_
↪fragmaps_2xyz
$SILCSBIODIR/utils/rename_fragmaps.sh silcs_fragmaps_1abc silcs_fragmaps_
↪2xyz 1abc_fixed 2xyz_fixed
```

12.5 Protein–Protein Interaction Maps

Protein–protein interactions (PPI) modulate the function and activity of proteins in a variety of biological processes. Thus, understanding PPIs can facilitate the identification of targets for the development of both small-molecule and protein-based therapeutics. Protein–protein complex structures provide atomistic level insights into PPIs that drive complex formation, which can be used to guide protein engineering or the design of therapeutic molecules to enhance or diminish specific PPI for a desired pharmacological outcome. SilcsBio provides a utility to study PPIs through the command line interface, SILCS-PPI. SILCS-PPI uses SILCS FragMaps to efficiently guide fast Fourier transform (FFT)-based protein docking. The resulting SILCS-PPI docking provides probability distributions of PPI interactions over the surface of both partner proteins, allowing for the identification of alternate binding poses. The SILCS-PPI utility can additionally be used to build PDB structures of protein–protein complexes encompasing these alternate binding poses. Details on SILCS-PPI are provided in ref [26]. The procedure for performing SILCS-PPI on two partner proteins is provided below:

1. Set up and run SILCS-PPI calculations:

   ```
   $SILCSBIODIR/ppi/1_setup_ppi prot1=<first prot pdb> mapsdir1=
   ↪<location and
   name of directory containing FragMaps> prot2=<second prot pdb>␣
   ↪mapsdir2=<location
   and name of directory containing FragMaps>
   ```

   The command for setting up and running SILCS-PPI calculations requires precomputed FragMaps of the two proteins of interest. If self-interactions are of interest, as in the case of protein aggregation, then the same input should be provided for `prot1` and `prot2` as well as `mapsdir1` and `mapsdir2`.

   Required parameters:

   - Path and name of first input protein PDB file (receptor protein):

     ```
     prot1=<first prot pdb>
     ```

   - Path and name of FragMaps for first input protein (receptor protein):

     ```
     mapsdir1=<location and name of directory containing␣
     ↪FragMaps for first protein>
     ```

- Path and name of second input protein PDB file (ligand protein):

```
prot2=<second prot pdb>
```

- Path and name of FragMaps for second input protein (ligand protein):

```
mapsdir2=<location and name of directory containing␣
↪FragMaps for second protein>
```

Optional parameters:

- Option to use the full FragMap of protein 1 (receptor protein):

```
fullmap=<true/false; flag for using full map or not.␣
↪default=true>
```

The SILCS-PPI calculation can be focused on a portion of the receptor protein. When `fullmap=false`, the `center` and `radius` parameters must be set.

- Rotation step size:

```
rotsize=<rotation step size; default=15 degrees>
```

- Number of solutions saved per rotation:

```
numsol=<# of solution per rotation; default=1>
```

- Center for transformation:

```
center=<"x,y,z"; e.g. center="20,30,-2">
```

The `center` parameter defines the center of the protein–protein contact region on the receptor protein. The `center` parameter must be used with the `radius` parameter when `fullmap=false`.

- Radius for transformation:

```
radius=<integer value>
```

The `radius` parameter defines the radius of the protein–protein contact region on the receptor protein. All solutions will be restricted to be within the radius from the specified center. The `radius` parameter must be used with the `center` parameter when `fullmap=false`.

- Option to bundle jobs:

```
bundle=<true/false; default=false>
```

- Number of jobs to bundle into one larger job:

```
njobs=<# of jobs used when bundle=true; default=16>
```

- Option to filter energy grids:

```
filter=<true/false; default=true>
```

By default, energy grids are filtered prior to SILCS-PPI calculations to remove unnecessary information. In the FragMaps, all voxels that are 5 Å away from the excluded area (defined by exclusion maps) are removed as those grids correspond to bulk phase behavior of the solutes rather than binding patterns on the protein surface. In the protein probability grids (PPGs), all voxels that are overlapping with the excusion map are removed as they represent the repulsive core of the protein.

2. Collect SILCS-PPI results:

```
$SILCSBIODIR/ppi/2_collect_ppi prot1=<first prot pdb> prot2=<second␣
↪prot pdb>
```

Required parameters:

- Path and name of first input protein PDB file (receptor protein):

```
prot1=<first prot pdb>
```

- Path and name of second input protein PDB file (ligand protein):

```
prot2=<second prot pdb>
```

Note: The entries for `prot1` and `prot2` must be the same as those used in the first step, `1_setup_ppi`.

Optional parameters:

- Path and name of SILCS-PPI results:

```
ppidir=<folder where PPI result is stored; default: 3_ppi>
```

- Option to cluster results:

```
cluster=<perform clustering; default=true>
```

- Number of clusters to output:

```
topn=<number of top cluster to output; default=2000>
```

3. Build protein–protein complex structures in PDB format:

```
$SILCSBIODIR/ppi/3_build_cmpx_pdb prot1=<receptor prot pdb> prot2=
↪<ligand prot pdb>
logfile=<PPI logfile> prefix=<prefix for output>
```

Required parameters:

- Path and name of first input protein PDB file (receptor protein):

```
prot1=<first prot pdb>
```

- Path and name of second input protein PDB file (ligand protein):

```
prot2=<second prot pdb>
```

Note: The entry for `prot1` and `prot2` must be the same as those used in the first step, `1_setup_ppi`.

- Path and name of SILCS-PPI log file:

```
logfile=<PPI logfile>
```

- Prefix for output PDB file names:

```
prefix=<prefix for output>
```

12.6 Difference FragMaps

The difference between SILCS FragMaps of structurally similar proteins can provide insights into differences in their activities. The SilcsBio software provides a utility in the command line interface (CLI) to calculate the difference between two SILCS FragMaps. The resulting difference FragMaps are a powerful tool to qualitatively analyze the difference between the functional group affinity pattern on two similar proteins including, but not limited to, kinases, BCL2 proteins, and mutants of the same protein.

A qualitiative analysis to investigate the differences in similar proteins may be accomplished by visualizing the difference FragMaps, which will offer insights into regions of the protein more favorable or less favorable for certain functional groups. Such information may be used to design molecules highly specific for only one protein (or dual specific for both proteins) or facilitate the identification and design of agonists versus antagonists, such as for GPCRs. See references [32] or [21] for example applications of difference FragMaps. The two-step procedure for calculating difference FragMaps is described below.

1. Copy the two `silcs_fragmaps_<protein PDB>` directories you wish to use to a new directory in which the difference calculations will be performed.

   Please make sure that the FragMaps meet *one* of the following criteria:

   - Both sets of FragMaps have the same grid setup, which means the header of the `silcs_fragmaps_<protein PDB>/maps/*.map` files are the same.
   - Both sets of FragMaps have were created relative to proteins having the same rotational and translational alignments. If the input proteins were not aligned to each other prior to running SILCS, you must use the **"ref="** option with the `2b_gen_maps` command to ensure alignment. A map cutting algorithm will be used to make the FragMaps compatible, with the smaller dimensions of the two grids used to calculate the difference FragMaps.

   Warning: It is imperative that the two SILCS FragMaps used to calculate difference FragMaps are oriented to structurally aligned proteins. Difference FragMaps that result from incompatible FragMaps will not provide any useful insights.

2. Generate the difference FragMaps using the following command:

```
$SILCSBIODIR/utils/calc_difference_maps.sh <silcs_fragmaps_#1>
↪<silcs_fragmaps_#2>
```

   The difference FragMaps will be saved in a directory `DIFF_MAPS_<fragmap#1>_VS_<fragmap#2>` and can be visualized in the same way as standard FragMaps (*Visualizing SILCS FragMaps*).

Tip: The difference in GFE values are calculated as `<silcs_fragmap_#1>` - `<silcs_fragmap_#2>`. In the difference FragMaps, if the value at a grid point is negative, the grid free energy (GFE) of the grid point in `<silcs_fragmap_#1>` is more favorable than in `<silcs_fragmap_#2>` and if it is positive, then `<silcs_fragmap_#2>` is more favorable than `<silcs_fragmap_#1>`.

Note: The difference FragMaps are intended *ONLY* for visualization and *NOT* for any quantitative calculations.

12.7 Average FragMaps

The average between SILCS FragMaps of structurally similar proteins can be used to identify ligands that bind bind to those proteins. The SilcsBio software provides a utility in the command line interface (CLI) to calculate the average over two or more sets of SILCS FragMaps. Average FragMaps may be used for a qualitative understanding of regions of the proteins that will interact similarly with certain functional groups. Average FragMaps may also be used quantitatively, with caution, for re-scoring of SILCS-MC docked ligands. The two-step procedure for calculating average FragMaps is described below.

1. Copy all the `silcs_fragmaps_<protein PDB>` directories you wish to use to a new directory in which the average calculations will be performed.

   Please make sure that the FragMaps meet *one* of the following criteria:

   - All sets of FragMaps have the same grid setup, which means the header of the `silcs_fragmaps_<protein PDB>/maps/*.map` files are the same.
   - All sets of FragMaps have were created relative to proteins having the same rotational and translational alignments. If the input proteins were not aligned to each other prior to running SILCS, you must use the **"ref="** option with the `2b_gen_maps` command to ensure alignment. A map cutting algorithm will be used to make the FragMaps compatible, with the smaller dimensions of the two grids used to calculate the difference maps.

   Warning: It is imperative that the two SILCS FragMaps used to calculate average FragMaps are oriented to structurally aligned proteins. Average FragMaps that result from incompatible FragMaps will not provide any useful insights.

2. Generate the average FragMaps using the following command:

   ```
   $SILCSBIODIR/utils/calc_average_maps.sh <path-to-directory-with-
   ↪all-silcs_fragmaps_*> <silcs_fragmaps_ref>
   ```

   The command will require two inputs:

   - Path and name of the directory containing all SILCS FragMaps to be averaged (`<path-to-directory-with-all-silcs_fragmaps_*>`). This directory corresponds to the "new directory" created in the first step.
   - Path and name of the SILCS FragMaps directory containing the reference PDB structure (`<silcs_fragmaps_ref>`).

   The average FragMaps will be saved in a directory `AVG_MAPS` and can be visualized in the same way as standard FragMaps (*Visualizing SILCS FragMaps*). When visualizing

the average FragMaps, the FragMaps will be overlaid onto the reference PDB structure specified in the second step.

Note: The FragMaps *MUST* be for highly homologous proteins and *MUST* be generated by using the most common structure as reference for the structural alignments.

12.8 4D Bioavailability (4DBA) Calculation

The bioavailability of compounds can be evaluated using 4D Bioavalability (4DBA) scores. Compounds with higher 4DBA scores are more likely to have oral drug-like characteristics. More details can be found in reference [33]. The 4DBA scores may facilitate lead optimization studies by indicating which compounds should be prioritized for chemical synthesis and biological testing.

The 4DBA score is calculated using four descriptors:

1. MW (Molecular Weight)
2. log P(o/w) (Logarithm of the octanol-water partition coefficient)
3. HDO (Number of hydrogen bond donors)
4. HAC (Number of hydrogen acceptors)

The SilcsBio software provides a utility through the command line interface (CLI) to calculate the four descriptors (MW, log P(o/w), HDO, HAC) and 4DBA scores. The 4DBA score can be calculated using the following command:

```
python $SILCSBIODIR/utils/python/calc_4dba.py --database <ligand SD file>␣
↪--method <method for descriptors>
```

The `calc_4dba.py` code requires the following arguments:

- Database of compounds in SD file format:

  ```
  --database <ligand SD file>
  ```

  The four descriptors and 4DBA scores will be calculated for all ligands in the SD file.

- Method used to calculate the descriptors (optional):

  ```
  --method <method for descriptors; rdkit or moe;␣
  ↪default=rdkit>
  ```

  Using the `--method rdkit` option will use RDKit to calculate the descriptors. By default the method will be set to `rdkit`.

Using the `--method moe` option will use MOE to calculate the descriptors. This requires that the database provided in sdf format has been saved using MOE after calculating the descriptors.

- Output file name (optional):

```
--output <output file name; default=lgfe-4dba.csv>
```

The output is a CSV file with the 4DBA score for each compound in the database. By default the output file name will be `lgfe-4dba.csv`.

12.9 Analysis of Ligand Substructures

The SilcsBio software provides a utility through the command line interface (CLI) to calculate the properties of substructures within SILCS-MC docked ligands.

```
python $SILCSBIODIR/utils/python/delta_lgfe_analysis.py --minconfpdb
↪<minconfpdb directory>
```

Required parameters:

- Path to directory containing SILCS-MC docked ligands:

```
--minconfpdb <minconfpdb directory>
```

- Path to parent structure SD file:

```
--parent <parent molecule SD file>
```

Optional parameters:

- Path to output CSV file:

```
--output <output csv file>
```

- Time out limit for each ligand:

```
--timeout <maximum time (in seconds) allowed for the␣
↪computation of the Maximum Common Substructure (MCS)␣
↪across all molecules; default=30s>
```

After running the above command, a csv (`delta_lgfe_analysis.csv` by default) will be created. The csv file will list the

1. ligand name (`Ligand`, first column),
2. total LGFE (`LGFE`, second column),

3. LE (`LE`, third column),
4. change in total LGFE from the parent ligand (`dLGFE`, fourth column),
5. SMILES of the common substructure (`CSS SMILES`, fifth column),
6. change in LGFE contribution of the common substructure (`CSS dLGFE`, sixth column),
7. SMILES of the modification or different substructure (`DSS SMILES`, seventh column),
8. change in LGFE contribution by the modification or different substructure (`DSS dLGFE`, eight column), and
9. RMSD of the common substructure in reference to the parent ligand (`CSS RMSD`, nineth column).

The output csv file is compatible with cheminformatics softwares such as MOE or Molsoft for viewing and further processing. Below is the output visualized in Molsoft with SMILES converted to 2D structures:

	Ligand	LGFE	LE	Δ LGFE	CSS SMILES	CSS Δ LGFE	DSS SMILES	DSS Δ LGFE	CSS RMSD
1	3fly_lig_638	-7.912	-0.316	0	smiles	-0	smiles	0	0
2	JOIN_20_21_acetamide_1079	-9.935	-0.343	-1.987	smiles	0.123	smiles	-2.11	0.266
3	JOIN_20_21_acetamide_2_917	-9.772	-0.337	-1.841	smiles	0.119	smiles	-1.96	0.673
4	JOIN_20_21_acetone_835	-8.683	-0.299	-0.734	smiles	0.916	smiles	-1.65	0.9
5	JOIN_20_21_ch2_cho_472	-8.888	-0.317	-0.979	smiles	0.031	smiles	-1.01	0.379
6	JOIN_20_21_ch2_cn_89	-7.975	-0.285	-0.064	smiles	0.076	smiles	-0.14	0.39
	JOIN_20_21_ch2_coo_853	-7.323	-0.253	0.597	smiles	0.777	smiles	-0.18	1.623

12.10 Plotting Probe Data

The SilcsBio software provides a utility to monitor probe concentrations and excess chemical potentials (μ_{ex}) as a function of GCMC-MD cycle. The following python script will generate these plots:

```
python $SILCSBIODIR/utils/python/plot_probe_data.py --prot <Protein PDB␣
↪File>

optional arguments:
  -h, --help            show this help message and exit
  --prot PROT           protein PDB file (Required)
  --numsys NUMSYS       number of runs (Default=10)
  --halogen HALOGEN     halgen probes True/False (Default=False)
  --begin BEGIN         first gcmc-md cycle (Default=1)
  --end END             number of gcmc-md cycles (Default=100)
  --vertical VERTICAL   plot together vertically/horizontally␣
↪(Default=False)
```

Below is an example of an output from running the python script:

```
1 BENX Acceptance: ins = 1.724% del = 1.771% trn = 15.665% rot = 2.013%
1 PRPX Acceptance: ins = 9.596% del = 10.202% trn = 24.742% rot = 8.106%
1 DMEE Acceptance: ins = 2.396% del = 2.438% trn = 18.753% rot = 2.162%
1 MEOH Acceptance: ins = 1.014% del = 1.037% trn = 13.012% rot = 0.680%
1 FORM Acceptance: ins = 0.772% del = 0.809% trn = 14.391% rot = 0.639%
1 IMIA Acceptance: ins = 1.183% del = 1.211% trn = 8.683% rot = 0.664%
1 ACEY Acceptance: ins = 1.453% del = 1.496% trn = 4.402% rot = 0.644%
1 MAMY Acceptance: ins = 3.865% del = 4.031% trn = 8.300% rot = 2.009%
1  SOL Acceptance: ins = 0.505% del = 0.506% trn = 10.704% rot = 1.026%
```

The output lists the SILCS simulation number, SILCS solute, and the accepted insertions (`ins`), deletions (`del`), translations (`trn`), and rotations (`rot`).

A detailed output for each simulation system will be stored in `status.<system number>.txt` text file in the `2a_run_gcmd` directory. An example of the contents of `status.<system number>.txt` is shown below:

```
 cycle fragname   curr_N   targ_N  percent        hfe     muexdx      ␣
↪vardx       dxdx       muex
   1     benx      118       98   120.41     -0.790      1.000      0.000␣
↪     0.000     -2.790
   1     prpx      118       98   120.41      1.960      0.500      0.000␣
↪     0.000      1.460
   1     dmee      118       98   120.41     -1.440      1.000      0.000␣
↪     0.000     -1.440
   1     meoh      118       98   120.41     -5.360      2.000      0.000␣
↪     0.000     -5.360
   1     form      118       98   120.41    -10.920      2.500      0.000␣
↪     0.000    -10.920
```

(continues on next page)

(continued from previous page)

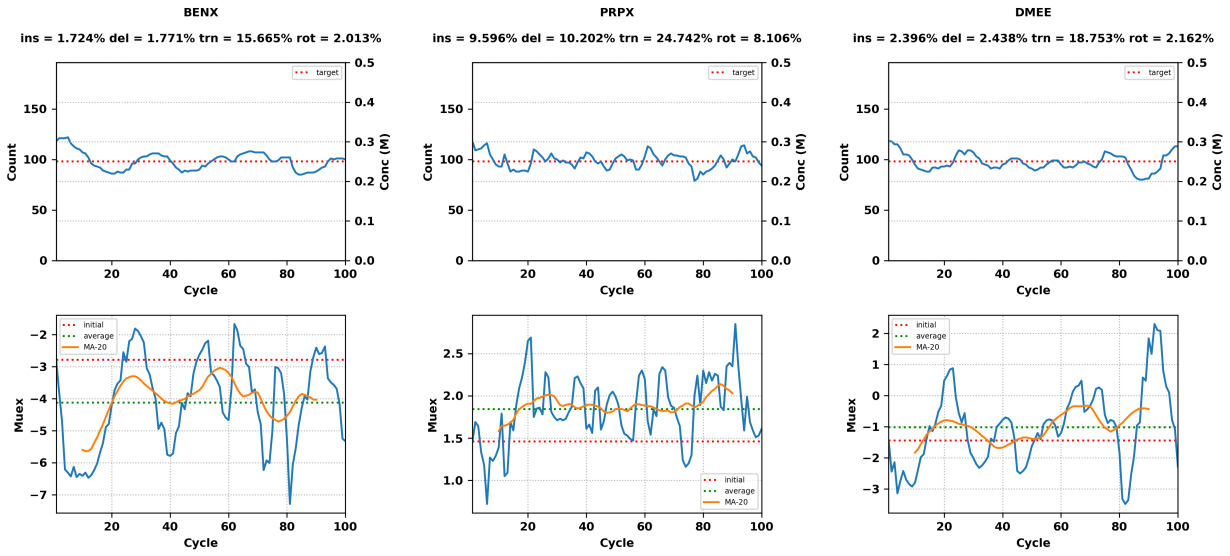
```
   1      imia       118        98    120.41     -14.180       3.000       0.000␣
↪      0.000     -14.180
   1      acey       118        98    120.41     -97.310      10.000       0.000␣
↪      0.000     -97.310
   1      mamy       118        98    120.41     -68.490       7.000       0.000␣
↪      0.000     -68.490
   1       sol     21750     21750    100.00      -5.600       0.500       0.000␣
↪      0.000      -5.600
   2      benx       121        98    123.47      -0.790       1.000      -0.920␣
↪     -0.300      -3.790
   2      prpx       109        98    111.22       1.960       0.500      -0.220␣
↪      0.450       1.690
   2      dmee       118        98    120.41      -1.440       1.000      -1.600␣
↪     -0.000      -2.440

...

  99      imia       100        98    102.04     -14.180       3.000      -0.240␣
↪      0.000      -7.160
  99      acey       109        98    111.22     -97.310      10.000       0.000␣
↪      1.000     -95.310
  99      mamy       104        98    106.12     -68.490       7.000       0.000␣
↪      0.700     -83.890
  99       sol     21818     21750    100.31      -5.600       0.500      -0.031␣
↪      0.000      -6.071
 100      benx       100        98    102.04      -0.790       1.000      -0.080␣
↪      0.000      -5.330
 100      prpx        94        98     95.92       1.960       0.500       0.080␣
↪      0.000       1.610
 100      dmee       113        98    115.31      -1.440       1.000      -1.200␣
↪     -0.000      -2.280
 100      meoh       107        98    109.18      -5.360       2.000      -0.720␣
↪     -0.400      -3.600
 100      form        98        98    100.00     -10.920       2.500       2.500␣
↪      0.000      -8.820
 100      imia       109        98    111.22     -14.180       3.000      -1.320␣
↪     -2.700     -11.180
 100      acey       106        98    108.16     -97.310      10.000      -3.200␣
↪      3.000     -95.510
 100      mamy       105        98    107.14     -68.490       7.000      -1.960␣
↪     -0.700     -86.550
 100       sol     21793     21750    100.20      -5.600       0.500      -0.020␣
↪      0.000      -6.090
```

In the `status.<system number>.txt` file, each row corresponds to a given solute molecule in a given cycle. In each row, the following information is provided:

- the cycle (`cycle`),
- SILCS solute (`fragname`),
- number of the solute molecule in the cycle (`curr_N`),
- targeted number of the solute molecule (`targ_N`),
- the percent number of the solute molecule over the targeted number of that solute molecule (`percent`),
- the approximate hydration free energy in kcal/mol of the solute obtained from free energy perturbation simulations (`hfe`) [9],
- the estimated magnitude of excess chemical potential variation (`muexdx`),
- the standard variation in excess chemical potential (`vardx`),
- the instantaneous variation in the excess chemical potential (`dxdx`), and
- the excess chemical potential (`muex`).

More information on GCMC cycles and each variable is available in reference [34].

In addition, the python script will produce plots as `.png` files for each probe in each simulation system. These plots will be stored in `2a_run_gcmd/acceptance_analysis/` as `count.<protein PDB name>.<system number>.png`. Below is an example of the output plots:



12.11 Format Convertor

The program, `format_convertor`, converts ligand files from one format to another format. It can also split a multiple-ligand SD file into single-ligand SD files.

The supported formats for formart conversion are:

1. SD, SDF
2. Mol, Mol2
3. PDB

Help menu:

```
$SILCSBIODIR/programs/format_convertor

    --split        [MODE] split: split input SDF file into a folder
    --convert      [MODE] convert: convert input file into other format
    --inf FILE     *[INPUT] input file name
    --outf FILE    *[OUTPUT] output file/folder name
    --iformat arg  *[FORMAT] input file format: PDB, MOL2, SDF, MOL
    --oformat arg  *[FORMAT] out file format: PDB, MOL2, SDF, MOL
-v, --verbose      verbose output
-h, --help         print help
```

Example usage for spliting one multiple-ligand SD file into single-ligand SD files:

```
$SILCSBIODIR/programs/format_convertor --split --inf i.sdf --outf dir
```

Example usage for converting a single-ligand SD file to a PDB file:

```
$SILCSBIODIR/programs/format_convertor --convert --inf i.sdf --outf o.pdb␣
↪--iformat SDF --oformat PDB
```

12.12 Biologics Formulation Series Generator

The occupancy analysis (`occupancy_xls.py`) in the final steps of the SILCS-Biologics workflow require an input file (`expt_data.csv`) defining the combination and concentrations of excipients in a given set of formulations to be analyzed. If the user wishes to analyze a range of formulations with varying excipients and concentrations, SilcsBio provides the utility to generate an input file that systematically varyies the excipients and concentration of each formulation.

The following command will generate the input file for `occupancy_xls.py`:

```
python $SILCSBIODIR/utils/python/excipients/create_expt_data.py \
     --in_file <IN_FILE> --n_excip <N_EXCIP> --out_file <OUT_FILE> --
↪advanced_help
```

Optional parameters:

- Name of input file specifying excipients

  ```
  --in_file <IN_FILE>
  ```

  If `--in_file` is not specified, then the default, `specs.csv`, will be used. Below is an example of the contents of `specs.csv`:

  ```
  Molecule,Type
  arginine,excipient
  sucrose,excipient
  alanine,excipient
  succinate,buffer
  histidine,buffer
  ```

- Maximum number of excipients in a single formulation

  ```
  --n_excip <N_EXCIP>
  ```

  If `--n_excip` is not specified, then the maximum number of excipients in a given formulation will be set to 1. Values greater than 4 are not currently supported.

- Output file to be used in occupancy calculations

  ```
  --out_file <OUT_FILE>
  ```

  If `--out_file` is not specified, then the output file will be named `expt_data.csv`. This file can be directly used as input in `occupancy_xls.py` for formulation analysis.

Advanced parameters:

- Show all advanced options.

  ```
  --advanced_help
  ```

- Concentration of excipients to be analyzed in mM:

  ```
  ----excipient_conc <EXCIPIENT_CONC>
  ```

  By default, the array of excipient concentrations is `[0, 10, 25, 50, 100, 150, 200, 250, 500]`.

- Concentration of buffers to be analyzed in mM:

```
----buffer_conc <BUFFER_CONC>
```

By default, the array of buffer concentrations is `[20]`. Note that by default, the buffer concentration will only be evaluated at 20 mM.

- Concentration of other molecules to be analyzed in mM:

```
----other_conc <OTHER_CONC>
```

By default, the array of buffer concentrations is `[0.01]`. Note that by default, the buffer concentration will only be evaluated at 0.01 mM.

12.13 Protein Effective Charge Estimator

Protein charge contributes to a range of solution behaviors including protein aggregation and solution viscosity. The charge of a protein is a system property, dependant on environmental factors such as pH, buffer composition, and ionic strength. Conventionally, protein charge is computationally estimated using sequence data and a set of pKa values for proton dissociation from ionizable groups. However, this does not account for the contributions of bound solute molecules and ions. The SilcsBio software provides the utility to estimate the effective charge of a protein accounting for bound excipients, buffers, and/or ions. Currently, the monoatomic ions supported are sodium, potassium, chloride, fluoride, bromide, and iodide.

1. **Run SILCS-Hotspots calculations with ion-specific params-file**:

   To estimate the effective charge of a protein, the user must first identify binding sites of the excipients, buffers, and/or ions in the solution using the first two steps of SILCS-Hotspots (`1_setup_silcs_hotspots` and `2_collect_hotspots`). Users may follow the instructions outlined in *SILCS-Hotspots*. Note that SILCS FragMaps of the protein of interest must be generated to run SILCS-Hotspots.

   If the ions under investigation include potassium, fluoride, bromide, iodide, nitrate, or thiocyanate, a different template parameters file is required. The custom parameter file can be specified by adding `paramsfile=$SILCSBIODIR/utils/python/excipients/charge_est/params_ions.tmpl` to the `1_setup_silcs_hotspots` command.

2. **Calculate the estimated protein effective charge**:

   After running `2_collect_hotspots`, use the following command to calculate the protein effective charge for a given solution:

```
python $SILCSBIODIR/utils/python/excipients/charge_est/charge_calc.
↪py \
    --protein_charge <sequence-based protein charge> \
    --cation <cation name> --anion <anion name> \
```

(continues on next page)

(continued from previous page)

```
    --cation_conc <cation concentration (mM)> --anion_conc <anion␣
↪concentration (mM)>
```

Required parameters:

- Sequence-based charge of the protein:

```
--protein_charge <sequence-based protein charge>
```

  The sequence-based charge of the protein may be calculated from external software such as PropKa.

- Names of the cation and anion:

```
--cation <cation name>
--anion <anion name>
```

- Concentrations of the cation and anion in mM:

```
--cation_conc <cation concentration (mM)>
--anion_conc <anion concentration (mM)>
```

Optional parameters:

- Name of the buffer molecule:

```
--buf <buffer name>
```

- Name(s) of the excipient(s):

```
--excipient <excipient name>
--excipient_2 <second excipient name>
--excipient_3 <third excipient name>
```

  Note that the protein effective charge estimator is able to accomodate up to 3 excipients.

- Concentration of the buffer molecule:

```
--buffer_conc <buffer concentration>
```

- Concentration of the excipient(s):

```
--excipient_conc <excipient concentration>
--excipient_2_conc <second excipient concentration>
--excipient_3_conc <third excipient concentration>
```

- Path and name of the SILCS-Hotspots directory:

```
--hotspots_dir <path and name of hotspots directory>
```

By default, `--hotspots_dir` will be set to `4_hotspots`.

- Charge of the cation:

```
--cation_charge <cation charge>
```

By default, `--cation_charge` will be set to `1`.

- Charge of the anion:

```
--anion_charge <anion charge>
```

By default, `--anion_charge` will be set to `-1`.

- Charge of the buffer:

```
--buffer_charge <buffer charge>
```

By default, `--buffer_charge` will be set to `0`.

- Charge of the excipient(s):

```
--excipient_charge <excipient charge>
--excipient_2_charge <second excipient charge>
--excipient_3_charge <third excipient charge>
```

By default, `--excipient_charge`, `--excipient_2_charge`, and `--excipient_3_charge` will be set to `0`.

- Occupancy cutoff of the buffer:

```
--bufocc <buffer occupancy cutoff>
```

By default, `--bufocc` will be set to `0.9`.

- Occupancy cutoff of the excipient(s):

```
--excocc <excipient occupancy cutoff>
```

By default, `--excocc` will be set to `0.9`. This cutoff will be applied to all excipients.

- Occupancy cutoff of the cation:

```
--catocc <cation occupancy cutoff>
```

By default, `--catocc` will be set to `0.70` for sodium or `0.85` for potassium. If the cation is not sodium or potassium, then `--catocc` must be specified.

- Occupancy cutoff of the anion:

```
--anocc <anion occupancy cutoff>
```

  By default, `--anocc` will be set to `0.70` for chloride, `0.75` for fluoride, `0.67` for bromide, `0.60` for iodide, `0.67` for nitrate, and `0.60` for thiocyanate. If the anion is none of the listed anions, then `--anocc` must be specified.

- Clustering radius for the cation:

```
--catrad <cation clustering radius>
```

  By default, `--catrad` will be set to `2.75` for sodium or `3.25` for potassium. If the cation is not sodium or potassium, then `--catocc` must be specified.

- Clustering radius for the anion:

```
--anrad <anion clustering radius>
```

  By default, `--anrad` will be set to `3.75` for chloride, `3.50` for fluoride, `4.25` for bromide, `4.50` for iodide, `3.75` for nitrate, and `4.50` for thiocyanate. If the anion is none of the listed anions, then `--anrad` must be specified.

CHAPTER

THIRTEEN

VIDEO TUTORIALS

13.1 Remote Server Setup on GUI

https://youtu.be/wjr7IKwvCEc

13.2 Visualizing SILCS FragMaps and Ligand Analysis on GUI

https://youtu.be/_WGLYXSBgtY

13.3 Running SILCS Simulations on GUI

https://youtu.be/R-_w8i9r-ps

13.4 SILCS-Hotspots on GUI

https://youtu.be/jFeKTiezznM

13.5 SILCS-MC Pose Refinement on GUI

https://youtu.be/F-G0_6JlNFo

13.6 SILCS-MC Ligand Optimization on GUI

https://youtu.be/o7cbDHxbm2o

13.7 SILCS-Pharm on GUI

https://youtu.be/4ZbIgQzEqP4

13.8 View/Edit Pharmacophore Features on GUI

https://youtu.be/Dm2wEh9s6KI

13.9 View/Modify Ligands on GUI

Coming Soon!

CHAPTER

FOURTEEN

FREQUENTLY ASKED QUESTIONS

14.1 Installation and Setup

14.1.1 How do I setup password-less entry for my remote server?

To setup password-less entry for your remote server, you need to copy the ssh-key from your local machine to the remote server. Here are the steps:

1. See if you have an existing public/private key pair on your local machine:

```
ls -al ~/.ssh/id_rsa*
```

The public key will be named `id_rsa.pub` and the private key will be named `id_rsa`. If you do not see these files, you do not have a public/private key pair.

2. If you do not have an existing public/private key pair, generate one on your local machine using the following command:

```
ssh-keygen -t rsa -b 4096
```

This command will generate a public/private key pair in the `~/.ssh` folder of your local machine.

3. Copy the public key to the remote server:

```
ssh-copy-id username@remote_server_ip_or_url
```

This command will copy the public key to the remote server and add it to the `~/.ssh/authorized_keys` file. You will be prompted to enter your password for the remote server. Once you enter the password, the ssh-key will be copied to the remote server. Next time you login to the remote server, you will not be prompted for a password.

4. (Optional) Create a configuration file for easy access:

   You can build a configuration file in the `~/.ssh` folder to store the connection information for the remote server. This will allow you to connect to the remote server using

a simple command like `ssh remote_server`. Here is an example of a configuration file ``**~/.ssh/config**`` :

```
Host <remote_server_name>
    Hostname <remote_server_ip_or_url>
    User <username>
    Port 22
    IdentityFile ~/.ssh/id_rsa
```

14.1.2 I installed the software, how do I test if it is correctly installed?

Because different users have different settings and requirements for their clusters or workstations, we provide a general job handling script for you to customize to your needs.

To assist with job handling script customization, example input files are available under the `$SILCSBIODIR/examples` folder.

For SILCS, use the following commands to make sure the software is correctly installed and the job handling script is working. If you are only interested in SSFEP simulations, you may skip to the SSFEP section below.

```
mkdir -p test/silcs
cd test/silcs
cp $SILCSBIODIR/examples/silcs/p38a.pdb .
$SILCSBIODIR/silcs/1_setup_silcs_boxes prot=p38a.pdb
$SILCSBIODIR/silcs/2a_run_gcmd prot=p38a.pdb numsys=1 nproc=1
```

If this set of commands runs without error, confirm that the SILCS job is running with the `check_progress` command:

```
$SILCSBIODIR/silcs/check_progress
```

and then go ahead and stop the successfully running SILCS job:

```
$SILCSBIODIR/silcs/2a_run_gcmd cancel=true sys=1
```

and confirm it is stopped:

```
$SILCSBIODIR/silcs/check_progress
```

Otherwise, if you experienced an error with the `1_setup-silcs_boxes` step, the software is not correctly installed, whereas if you experienced an error with the `2a_run_gcmd` step, the job handling scripts need to be edited. The job handling scripts for SILCS are:

- `templates/silcs/job_mc_md.tmpl`
- `templates/silcs/job_gen_maps.tmpl`

- templates/silcs/pymol_fragmap.tmpl
- templates/silcs/vmd_fragmap.tmpl
- templates/silcs/job_cleanup.tmpl

Typically the header portion of a job handling script requires editing. Please contact support@silcsbio.com if you need assistance.

For SSFEP, use the following commands to make sure the software is correctly installed and the job handling script is working.

```
mkdir -p test/ssfep
cd test/ssfep
cp $SILCSBIODIR/examples/ssfep/* .
$SILCSBIODIR/ssfep/1_setup_ssfep prot=4ykr.pdb lig=lig.mol2
$SILCSBIODIR/silcs/2_run_md_ssfep prot=4ykr.pdb lig=lig.mol2 nproc=1
```

If this set of commands runs without error, confirm that the SSFEP job is running with the `check_progress` command:

```
$SILCSBIODIR/ssfep/check_progress
```

and then go ahead and stop the successfully running SSFEP job:

```
$SILCSBIODIR/ssfep/2_run_md_ssfep cancel=true target=lig
$SILCSBIODIR/ssfep/2_run_md_ssfep cancel=true target=prot
```

and confirm it is stopped:

```
$SILCSBIODIR/ssfep/check_progress
```

Otherwise, if you experienced an error with the `1_setup_ssfep` step, the software is not correctly installed, and if you experienced an error with the `2_run_md_ssfep` step, the job handling script needs to be edited. The job handling scripts for SSFEP are:

- templates/ssfep/job_lig_md.tmpl
- templates/ssfep/job_prot_lig_md.tmpl
- templates/ssfep/job_dG.tmpl

Typically the header portion of a job handling script requires editing. Please contact support@silcsbio.com if you need assistance.

14.1.3 I don't have a cluster but I have a GPU workstation. What can I do?

You may be able to practically run the SilcsBio software if your GPU workstation has sufficient resources. An appropriate workstation may have at least 24 CPU cores, 4 GPUs, 64 GB of RAM, and 10 TB of disk space. Installing a job queueing system, such as the Slurm Workload Manager, will allow the SilcsBio server software to run on the workstation.

The SilcsBio Workstation is a turn-key GPU workstation hardware+software solution developed by SilcsBio that comes with all necessary software pre-installed. The SilcsBio workstation has a quiet, sleek form factor for use in an office setting and comes ready to plug in to a standard wall electrical socket. Please contact info@silcsbio.com for details.

14.1.4 I compiled my GROMACS with MPI and my job is not running

Please contact us so we can repackage the files with the appropriate command using `mpirun` instead.

Alternatively, you may edit the job handling script to use the appropriate GROMACS command.

For example, the `mdrun` command is specified at the top of `templates/ssfep/job_lig_md.tmpl` file:

```
mdrun="${GMXDIR}/gmx mdrun -nt $nproc"
```

You may edit this to

```
mdrun="mpirun -np $nproc ${GMXDIR}/gmx mdrun"
```

14.1.5 GROMACS on the head node does not run because the head node and compute node have different operating systems

In this case, we recommend compilng GROMACS on the head node and compiling `mdrun` only on the compute node.

Building only `mdrun` can be done by supplying the `-DGMX_BUILD_MDRUN_ONLY=on` keyword to the `cmake` command in the build process. Once the `mdrun` program is built, place it in the same $GMXDIR folder. Now template files needs to be edited to use the `mdrun` command properly on the compute node.

For example, the `mdrun` command is specified at the top of the `templates/ssfep/job_lig_md.tmpl` file:

```
mdrun="${GMXDIR}/gmx mdrun -nt $nproc"
```

You may edit this to

```
mdrun="${GMXDIR}/mdrun -nt $nrpoc"
```

14.1.6 I get the "error while loading shared libraries: libcudart.so.8.0: cannot open shared object file: No such file or directory" message during my setup

If you encounter this error, the most likely reason is that GROMACS was compiled on a machine having a GPU whereas the current machine where the command is being executed does not have a GPU.

It may be possible that the necessary library is already available for the machine even though it does not have a GPU. So, check if the `libcudart.so` file exists on the current machine. The most likely place is `/usr/local/cuda/lib64`. If the file exists in that location, add that path to your `LD_LIBRARY_PATH` environment variable..

If the library is not available on the current machine, we recommend following the *previous FAQ entry* to compile GROMACS and `mdrun` separately.

14.1.7 I get the "error while loading shared libraries: libssl.so.1.1: cannot open shared object file: No such file or directory"

If you encounter this error, that means your operating system needs to install Secure Sockets Layer / Transport Layer Security (SSL/TLS) library (>= version 1.1.0).

If you are using Debian-based Linux: Ubuntu, etc:

```
sudo apt-get update
sudo apt-get install libssl-dev
```

If you are using RedHat-based: Fedora, CentOS, Amazon Linux 2, etc:

```
sudo yum update -y
sudo yum -y install openssl11-libs.x86_64
```

14.2 SILCS Simulation Setup

14.2.1 I want to modify the force field and topology files for SILCS simulation

As an example, if there is the need to add extra bonds that are not present in the standard force field definitions, this is the procedure to make the necessary modifications. Please refer to the GROMACS documentation regarding modifying the .top and ffbonded.itp files.

First, run the following command. This will copy the basic force field to and generate an initial topology file in `1_setup/`, allowing you to edit them.

```
$SILCSBIODIR/silcs/1_setup_silcs_boxes prot=<prot PDB>
```

Then edit the force field parameter file `1_setup/charmm36.ff/ffbonded.itp`.

If you want to modify the topology file (e.g. to add a bond between two atoms), copy `1_setup/<prot>_gmx.top.1.bak` to `1_setup/<prot>_gmx.top`. Then edit `<prot>_gmx.top` and add the desired change under the appropriate list (e.g., add a bond between two atoms in the `[ bond ]` list).

Once the files are edited, re-run the 1_setup command with the `skip_pdb2gmx=true` keyword. This will preserve your edits and create the necessary files to run the SILCS simulations.

```
$SILCSBIODIR/silcs/1_setup_silcs_boxes prot=<prot PDB> skip_pdb2gmx=true
```

Once this completes, run the `$SILCSBIODIR/silcs/2a_run_gcmd` script to initiate your SILCS simulations.

14.2.2 How do I handle phosphorylated amino acids?

The following phosphorylated amino acids are supported:

- pSer
- pThr
- pTyr

To create a phosphorylated amino acid, rename that amino acid in your input pdb file as follows:

- SER => SP1 or SP2
- THR => THP1 or THP2
- TYR => TP1 or TP2

The number at the end of the amino acid name refers to whether the phosphate group has mono- or divalent charge.

14.2.3 What if my protein has a glycan attached to it?

While setting up a glycan-containing protein directly from a PDB file is not currently supported, you can set up your simulation system for SILCS if you have a PSF file created with the CHARMM36 force field.

An example can be found in the `$SILCSBIODIR/examples/glycan` folder. Running the `setup.sh` script in that directory will run the example and create a folder named `1_setup`.

For your own system, copy the `gromacs` folder and `setup.sh` file and edit the copied `setup.sh` file before running it:

```
psffile="psf/step1_pdbreader.psf" # PSF file
pdbfile="psf/step1_pdbreader.pdb" # PDB file
prefix="5vgp" # prefix for the SILCS simulation
```

14.2.4 What happens when I set up SILCS simulations with an input structure containing a metal ion?

SILCS simulation supports a variety of metal ions, including calcium, copper, iron, magnesium, manganese, nickel, and zinc. If an ion in the input structure is located close to protein atoms (~ 3A), the setup script will automatically create covalent linkages between the metal ion and nearby protein residues so as to ensure the coordination structure is maintained throughout the SILCS GCMC/MD simulations.

14.2.5 My protein contains iron and I want to set a +3 charge state

By default, the SILCS setup assigns a +2 charge to iron. If you want to change the charge of the iron ion, change the residue name of the ion in the input PDB to FE3. The setup script will then assign a +3 charge to that ion.

14.2.6 How do I include a covalently bound ligand/cofactor in SILCS simulations?

SILCS-Small Molecule users with the CGenFF Suite license will be able to generate the necessary topology and parameter files as well as PDB structures needed to perform SILCS simulations of a protein covalently bound to a ligand/cofactor. For more information, please refer to *CGenFF for Covalent Ligands*.

SILCS-Small Molecule users without the CGenFF Suite license are still able to achieve a good approximation of the covalently bound ligand structure by using positional restraints to maintain relative geometries during the SILCS simulation. To do so, provide the ligand/cofactor as an individual ligand using the option **lig=lig.mol2** during setup. The mol2 file should contain only the

ligand/cofactor molecule and must have all hydrogen atoms and a appropriate three-dimensional coordinates that provide a reasonable internal geometry and place it correctly relative to the protein. Weak position restraints will be automatically added on the non-hydrogen atoms of ligand/cofactor, so it will be positionally restrained. (Note: you will also need to set the option **scramblesc=false** when running `1_setup_silcs_boxes` in order to keep protein sideschain conformations from being scrambled.)

```
$SILCSBIODIR/silcs/1_setup_silcs_boxes prot=<prot PDB> lig=lig.mol2␣
↪scramblesc=false
```

However, the amino acid sidechain to which the ligand/cofactor is meant to be covalently bound will be unrestrained. To address this, you will need to manually add weak position restraints on non-hydrogen atoms of the sidechain. To do this, edit the `1_setup/posre_protein_ca.itp` file which restraints C-alpha atoms and append it with the non-hydrogen atoms from your sidechain.

```
; position restraints for (atomname_CA_or_atomname_"C1'")

[ position_restraints ]
;  i funct       fcx        fcy        fcz
   5    1     50.208     50.208     50.208
  20    1     50.208     50.208     50.208
  46    1     50.208     50.208     50.208
  58    1     50.208     50.208     50.208
.
.
.
5571    1     50.208     50.208     50.208
5589    1     50.208     50.208     50.208
5603    1     50.208     50.208     50.208
5617    1     50.208     50.208     50.208
; **ADD YOUR SIDECHAIN ATOMS BELOW**
xxxx    1     50.208     50.208     50.208
xxxx    1     50.208     50.208     50.208
xxxx    1     50.208     50.208     50.208
xxxx    1     50.208     50.208     50.208
```

Please visualize the `1_setup/<prot>_silcs.1-10.pdb` files to make sure the ligand and sidechain are correctly prepared before you run the `2a_run_gcmd` command.

14.2.7 How do I fit my membrane protein in a bilayer as suggested by the OPM server?

1. Prepare your bare membrane protein with Alphafold, MOE etc. as <prot PDB>
2. Upload the <prot PDB> to the OPM server (PPM 3.0) and get the output.
3. Download the <OPM output pdb>, open it with PyMOL or VMD, and determine the required translation along the z-axis:

   PyMOL

   - Select protein atoms: `sele all and polymer`
   - Calculate center of mass (COM): `centerofmass sele`
   - Copy the **Z-coordinate_of_COM** and go to the next step

   VMD

   - Open VMD Main >> Extensions >> TkConsole
   - Create selection with protein atoms: `set sel1 [atomselect top "all and protein"]`
   - Calculate center of mass (COM): `set com1 [measure center $sel1 weight mass]`
   - Copy the **Z-coordinate_of_COM** and go to the next step
4. Align <prot PDB> to <OPM output pdb> for subsequent use OR extract protein from <OPM output pdb> for subsequent use.
5. Run setup with the additional `offset_z`:

```
$SILCSBIODIR/silcs-memb/1a_fit_protein_in_bilayer prot=<aligned prot␣
↪PDB> orient_principal_axis=false offset_z=<Z-coordinate_of_COM>
```

14.3 Fragmaps Usage and Visualization

14.3.1 I want to visualize FragMaps using MOE

By default, SILCS FragMaps are in the MAP grid file format. However, this file format is not supported in MOE. Please see *FragMaps in MOE* for detailed instructions on creating FragMaps in a MOE-compatible format.

14.3.2 I have a set of FragMaps generated by version 2021, can I use it in later version?

Yes, you can use the SILCS FragMaps generated by version 2021 or earlier in later versions of silcsbio.

You need to run the following command to make the FragMaps compatible with the latest version of silcsbio:

```
$SILCSBIODIR/utils/fragmaps_conversion.sh silcs_fragmaps_XXXX
```

14.3.3 I'm getting `NameError: name 'FragMap' is not defined` when I try to visualize FragMaps in PyMOL and I have a Apple M chip

This is a typical issue in the M-chip based macs as they have changed some architecture.

Here is the solution for this particular PyMol library issue.

1. Make a directory of library for libx11 (most likely your MacOS does not have this folder):

   ```
   sudo mkdir -p /usr/local/opt/libx11
   ```

2. Install XQuartz, if you don't have one.

   You can try to type `xquartz` in your `Launchpad`. If it cannot find this app, then you need to install it.

   If it can find this app, then skip this step.

   XQuartz official website: https://www.xquartz.org/

3. (Optional) Install libx11, if you don't have it.

   libx11 webpage: https://formulae.brew.sh/formula/libx11

   Installation command: brew install libx11

   You can check the target location "/opt/X11/", and the directory should be there and it should contain a subdirectory "lib".

4. Make a symbolic link.

   ```
   sudo ln -s /opt/X11/lib /usr/local/opt/libx11/lib
   ```

5. Start your PyMol and load the session file. You can either drag the session file and drop it to PyMol main window, or use the command to load the session file in pymol command.

CHAPTER

FIFTEEN

REFERENCES

Please refer to https://silcsbio.com/category/publications/ for a full list of publications featuring SilcsBio technology.

CHAPTER

SIXTEEN

ABBREVIATIONS

Table 16.1: List of Abbreviations

Abbreviation	Stands for
3D	Three-dimensional
4DBA	4D Bioavailability
ACS	Atom classification scheme
AWS	Amazon Web Services
BML	Bayesian machine learning
BSA	Buried surface area
CDIE	Constant dielectric scheme
CGenFF	CHARMM General Force Field
CHARMM	Chemistry at Harvard Macromolecular Mechanics
CLI	Command-line interface
COM	Center of Mass
FEP	Free energy perturbation
FGFE	Feature grid free energy
GCMC	Grand canonical Monte Carlo
GFE	Grid free energy
GUI	Graphical user interface
HPC	High performance computing
HTVS	High throughput virtual screening
LE	Ligand efficiency
LGFE	Ligand grid free energy
LJ	Lennard-Jones
MC	Monte Carlo
MD	Molecular dynamics

continues on next page

Table 16.1 – continued from previous page

Abbreviation	Stands for
MM	Molecular mechanics
MW	Molecular weight
PC	Percent correct
PPG	Protein probability grid
PPI	Protein–protein interaction
PPIP	Protein–protein interaction preference
R	Pearson correlation
RAA	Relative affinity analysis
RDIE	Distance dependent dielectric scheme
RGYR	Radius of gyration
RMSD	Root mean squared deviation
RMSF	Root mean squared fluctuation
SASA	Solvent accessible surface area
SILCS	Site Identification by Ligand Competitive Saturation
SM	Small-molecule
SSFEP	Single Step Free Energy Perturbation
VS	Virtual screening

CHAPTER

SEVENTEEN

CONTACT SUPPORT

Are you experiencing issues or need assistance or want to request a new feature?

—

Alternatively, you can copy the template below with relevant details and send it to **support@silcsbio.com**.

Providing detailed information will help us resolve your issue more efficiently.

—

Title:

(Short, clear summary of the issue/feature)

Expected Behavior:

(What should have happened?)

Actual Behavior:

(What actually happened?)

Steps to Reproduce:

1. 2. 3.

Environment:

- **Application / Version:**
- **Operating System:**

Other Relevant Details:

(Add any other information that might help us understand the issue.)

Error Messages / Logs:

(Copy-paste any error messages or logs, if available.)

Attachments:

(Screenshots, files, or links, if applicable.)

Severity (optional):

- [ ] Low
- [ ] Medium
- [ ] High
- [ ] Critical

—

Once you have completed the form, email it to **support@silcsbio.com**. We will get back to you as soon as possible.

BIBLIOGRAPHY

[1] Anastasia Croitoru, Sang-Jun Park, Anmol Kumar, Jumin Lee, Wonpil Im, Alexander D MacKerell Jr, and Alexey Aleksandrov. Additive CHARMM36 force field for nonstandard amino acids. *J. Chem. Theory Comput.*, 17(6):3554–3570, June 2021. doi: 10.1021/acs.jctc.1c00254.

[2] Samo Lesnik, Milan Hodoscek, Urban Bren, Christoph Stein, and Ana-Nicoleta Bondar. Potential energy function for fentanyl-based opiod pain killers. *J. Chem. Inf. Model*, 60(7):3566–3576, July 2020. doi: 10.1021/acs.jcim.0c00185.

[3] Oskar Klaja, James A Frank, and Ana-Nicoleta Bondar. Potential energy function for a photo-switchable lipid molecule. *J. Comput. Chem.*, 41(27):2336–2351, October 2020. doi: 10.1002/jcc.26387.

[4] You Xu, Kenno Vanommeslaeghe, Alexey Aleksandrov, Alexander D MacKerell Jr, and Lennart Nilsson. Additive CHARMM force field for naturally occurring modified ribonucleotides. *J. Comput. Chem.*, 37(10):896–912, April 2016. doi: 10.1002/jcc.24307.

[5] Alexander D MacKerell Jr, Sunhwan Jo, Sirish Kaushik Lakkaraju, Christoffer Lind, and Wenbo Yu. Identification and characterization of fragment binding sites for allosteric ligand design using the Site Identification by Ligand Competitive Saturation Hotspots approach (SILCS-Hotspots). *Biochim. Biophys. Acta. Gen. Subj.*, 1864(4):129519–129519, April 2020. doi: 10.1016/j.bbagen.2020.129519.

[6] Olgun Guvench and Alexander D MacKerell Jr. Computational Fragment-Based Binding Site Identification by Ligand Competitive Saturation. *PLoS Comput. Biol.*, 5(7):e1000435–12, July 2009. doi: 10.1371/journal.pcbi.1000435.

[7] Abhishek A Kognole, Anthony Hazel, and Alexander D MacKerell Jr. SILCS-RNA: Towards a structure-based drug design approach for targeting RNAs with small molecules. *J. Chem. Theory Comput.*, 18(9):5672–5691, July 2022. 10.1021/acs.jctc.2c00381.

[8] E Prabhu Raman, Wenbo Yu, Olgun Guvench, and Alexander D MacKerell Jr. Reproducing crystal binding modes of ligand functional groups using Site Identification by Ligand Competitive Saturation (SILCS) simulations. *J. Chem. Inf. Model.*, 51(4):877–896, April 2011. 10.1021/ci100462t.

[9] Sirish Kaushik Lakkaraju, E Prabhu Raman, Wenbo Yu, and Alexander D MacKerell Jr. Sampling of organic solutes in aqueous and heterogeneous environments using oscillating excess chemical potentials in grand canonical-like Monte Carlo-molecular dynamics simulations. *J. Chem. Theory Comput.*, 10(6):2281–2290, June 2014. doi: 10.1021/ct500201y.

[10] Erik B. Nordquist, Mingtian Zhao, Anmol Kumar, and Alexander D. MacKerell Jr. Combined Physics- and Machine-Learning-Based Method to Identify Druggable Binding Sites Using SILCS-Hotspots. *J. Chem. Inf. Model.*, 64(19):7743–7757, September 2024. doi: 10.1021/acs.jcim.4c01189.

[11] Yang Zhang and Jeffrey Skolnick. SPICKER: a clustering approach to identify near-native protein folds. *J. Comput. Chem.*, 25(6):865–871, April 2004. doi: 10.1002/jcc.20011.

[12] Richard D Taylor, Malcolm MacCoss, and Alastair D G Lawson. Rings in drugs. *J. Med. Chem.*, 57(14):5845–5859, July 2014. doi: 10.1021/jm4017625.

[13] Marc O'Reilly, Anne Cleasby, Thomas G Davies, Richard J Hall, R Frederick Ludlow, Christopher W Murray, Dominic Tisi, and Harren Jhoti. Crystallographic screening using ultra-low-molecular-weight ligands to guide drug design. *Drug Discov. Today*, 24(5):1081–1086, May 2019. doi: 10.1016/j.drudis.2019.03.009.

[14] Wenbo Yu, Sirish Kaushik Lakkaraju, E Prabhu Raman, Lei Fang, and Alexander D MacKerell Jr. Pharmacophore modeling using Site Identification by Ligand Competitive Saturation (SILCS) with multiple probe molecules. *J. Chem. Inf. Model.*, 55(2):407–420, February 2015. doi: 10.1021/ci500691p.

[15] E Prabhu Raman, Wenbo Yu, Sirish K Lakkaraju, and Alexander D MacKerell Jr. Inclusion of multiple fragment types in the Site Identification by Ligand Competitive Saturation (SILCS) approach. *J. Chem. Inf. Model.*, 53(12):3384–3398, December 2013. doi: 10.1021/ci4005628.

[16] Vincent D Ustach, Sirish Kaushik Lakkaraju, Sunhwan Jo, Wenbo Yu, Wenjuan Jiang, and Alexander D MacKerell Jr. Optimization and evaluation of Site-Identification by Ligand Competitive Saturation (SILCS) as a tool for target-based ligand optimization. *J. Chem. Inf. Model.*, 59(6):3018–3035, April 2019. doi: 10.1021/acs.jcim.9b00210.

[17] Paolo Tosco, Thomas Balle, and Fereshteh Shiri. Open3DALIGN: an open-source software aimed at unsupervised ligand alignment. *J. Comput. Aided Mol. Design*, 25(8):777–783, August 2011. doi: 10.1007/s10822-011-9462-9.

[18] Himanshu Goel, Anthony Hazel, Vincent D Ustach, Sunhwan Jo, Wenbo Yu, and Alexander D MacKerell Jr. Rapid and accurate estimation of protein–ligand relative binding affinities using site-identification by ligand competitive saturation. *Chem. Sci.*, 12(25):8844–8858, May 2021. doi: 10.1039/d1sc01781k.

[19] Anmol Kumar, Himanshu Goel, Wenbo Yu, Mingtian Zhao, and Alexander D Jr MacKerell. Modeling Ligand Binding Site Water Networks with Site-Identification by Ligand Competitive Saturation: Impact on Ligand Binding Orientations and Relative Binding Affinities. *ChemRxiv*, NA(NA):NA, March 2024. doi:10.26434/chemrxiv-2024-sqhw9.

[20] Geoffrey A Heinzl, Weiliang Huang, Wenbo Yu, Bennett J. Giardina, Yue Zhou, Alexander D MacKerell Jr, Angela Wilks, and Fengtian Xue. Iminoguanidines as allosteric inhibitors of the iron-regulated heme oxygenase (HemO) of Pseudomonas aeruginosa. *J. Med. Chem.*, 59(14):6929–6942, June 2016. doi: 10.1021/acs.jmedchem.6b00757.

[21] Maryanna E Lanning, Wenbo Yu, Jeremy L Yap, Jay Chauhan, Lijia Chen, Ellis Whiting, Lakshmi S Pidugu, Tyler Atkinson, Hala Bailey, Willy Li, Braden M Roth, Lauren Hynicka, Kirsty Chesko, Eric A Toth, Paul Shapiro, Alexander D MacKerell Jr, Paul T Wilder, and Steven Fletcher. Structure-based design of N-substituted 1-hydroxy-4-sulfamoyl-2-naphthoates as selective inhibitors of the Mcl-1 oncoprotein. *Eur. J. Med. Chem.*, 113:273–292, May 2016. doi: 10.1016/j.ejmech.2016.02.006.

[22] Wenbo Yu, David J. Weber, and Alexander D. Jr. MacKerell. Integrated Covalent Drug Design Workflow Using Site Identification by Ligand Competitive Saturation. *J. Chem. Theory Comput.*, 19(10):3007–3021, April 2023. doi:10.1021/acs.jctc.3c00232.

[23] László Petri, Attila Egyed, Dávid Bajusz, Tímea Imre, Anasztázia Hetényi, Tamás Martinek, Péter Ábrányi-Balogh, and György M. Keserű. An electrophilic warhead library for mapping the reactivity and accessibility of tractable cysteines in protein kinases. *Eur. J. Med. Chem.*, 207:112836, December 2020. 10.1016/j.ejmech.2020.112836.

[24] Wenbo Yu, David J. Weber, and Alexander D. Jr. MacKerell. Detection of Putative Ligand Dissociation Pathways in Proteins Using Site-Identification by Ligand Competitive Saturation. *J. Chem. Inf. Model.*, 65(6):3022–3034, March 2025. doi:10.1021/acs.jcim.4c01814.

[25] Wenbo Yu, Shashi Kumar, Mingtian Zhao, David J. Weber, and Alexander D. Jr. MacKerell. High-Throughput Ligand Dissociation Kinetics Predictions Using Site Identification by Ligand Competitive Saturation. *J. Chem. Theory Comput.*, 21(9):4964–4978, May 2025. doi:10.1021/acs.jctc.5c00265.

[26] Wenbo Yu, Sunhwan Jo, Sirish Kaushik Lakkaraju, David J Weber, and Alexander D MacKerell Jr. Exploring protein-protein interactions using the Site-Identification by Ligand Competitive Saturation methodology. *Proteins*, 87(4):289–301, April 2019. doi: 10.1002/prot.25650.

[27] Sunhwan Jo, Amy Xu, Joseph E Curtis, Sandeep Somani, and Alexander D MacKerell Jr. Computational characterization of antibody-excipient interactions for rational excipient selection using the site identification by ligand competitive saturation-biologics approach. *Mol. Pharm.*, 17(11):4323–4333, November 2020. doi: 10.1021/acs.molpharmaceut.0c00775.

[28] E Prabhu Raman, Kenno Vanommeslaeghe, and Alexander D MacKerell Jr. Site-specific fragment identification guided by Single-Step Free Energy Perturbation calculations. *J. Chem. Theory Comput.*, 8(10):3513–3525, October 2012. doi: 10.1021/ct300088r.

[29] E Prabhu Raman, Sirish Kaushik Lakkaraju, Rajiah Aldrin Denny, and Alexander D MacKerell Jr. Estimation of relative free energies of binding using pre-computed ensembles based on the Single-Step Free Energy Perturbation and the Site Identification by Ligand Competitive Saturation approaches. *J. Comput. Chem.*, 38(15):1238–1251, October 2016. doi: 10.1002/jcc.24522.

[30] K Vanommeslaeghe, E Hatcher, C Acharya, S Kundu, S Zhong, J Shim, E Darian, O Guvench, P Lopes, I Vorobyov, and Alexander D MacKerell Jr. CHARMM general force field: A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *J. Comput. Chem.*, 31(4):671–690, March 2010. doi: 10.1002/jcc.21367.

[31] Kenno Vanommeslaeghe and Alexander D MacKerell Jr. Automation of the CHARMM General Force Field (CGenFF) I: bond perception and atom typing. *J. Chem. Inf. Model.*, 52(12):3144–3154, December 2012. doi: 10.1021/ci300363c.

[32] Sirish Kaushik Lakkaraju, Wenbo Yu, E Prabhu Raman, Alena V Hershfeld, Lei Fang, Deepak A Deshpande, and Alexander D MacKerell Jr. Mapping functional group free energy patterns at protein occluded sites: nuclear receptors and G-protein coupled receptors. *J. Chem. Inf. Model.*, 55(3):700–708, March 2015. doi: 10.1021/ci500729k.

[33] Taiji Oashi, Ashley L. Ringer, E. Prabhu Raman, and Alexander D. MacKerell. Automated Selection of Compounds with Physicochemical Properties To Maximize Bioavailability and Druglikeness. *J. Chem. Inf. Model.*, 51:148–158, December 2011. doi: 10.1021/ci100359a.

[34] Mingtian Zhao, Abhishek A. Kognole, Sunhwan Jo, Aoxiang Tao, Anthony Hazel, and Alexander D. Jr. MacKerell. GPU-specific algorithms for improved solute sampling in grand canonical Monte Carlo simulations. *J. Comput. Chem.*, 44(20):1719–1732, July 2023. doi:10.1002/jcc.27121.