SilcsBio User Guide

Release 2019.1

November 2018

CONTENTS

SilcsBio LLC
8 Market Place, Suite 300
Baltimore, MD 21202
info@silcsbio.com
http://www.silcsbio.com

CHAPTER

ONE

ABOUT THIS DOCUMENT

SilcsBio offers a software suite for computer-aided drug design based on functional group mapping that includes capabilities for database screening, fragment-based drug design, and lead optimization. This documentation covers how to install and use the software.

CHAPTER

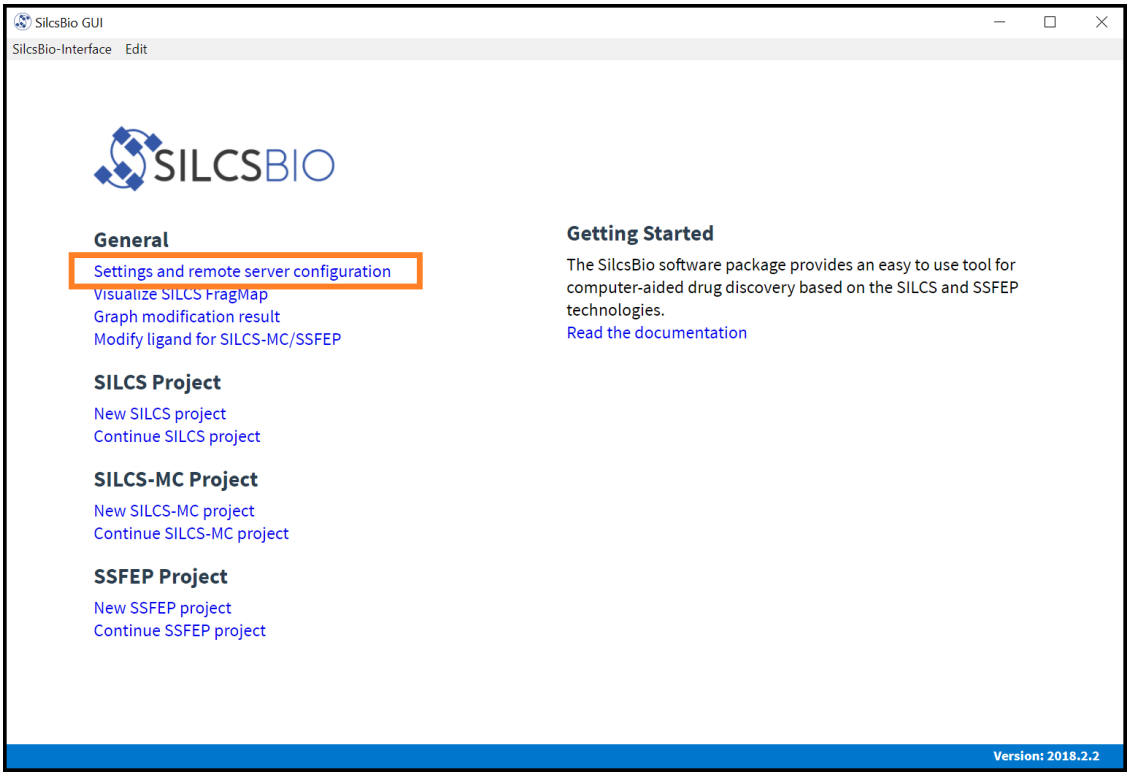
TWO

QUICKSTART GUIDE: GRAPHICAL USER INTERFACE

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

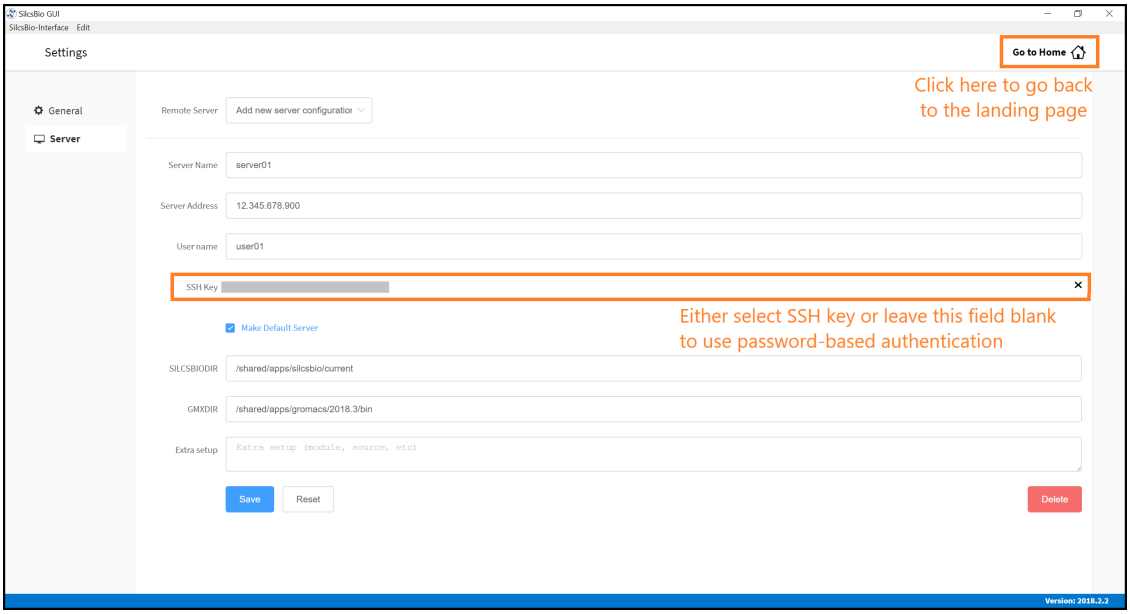
2.1 Remote server setup

The SilcsBio GUI is designed to work with the server installation of the SilcsBio software. Therefore, the remote server needs to be configured to properly use the GUI. When you launch the GUI, you will be presented with the Home page. From the Home page, select *Settings and remote server configuration*.



Within the "Settings" page, select *Server* and enter the remote server information, such as Server 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 remote server. Please contact support@silcsbio.com for assistance.

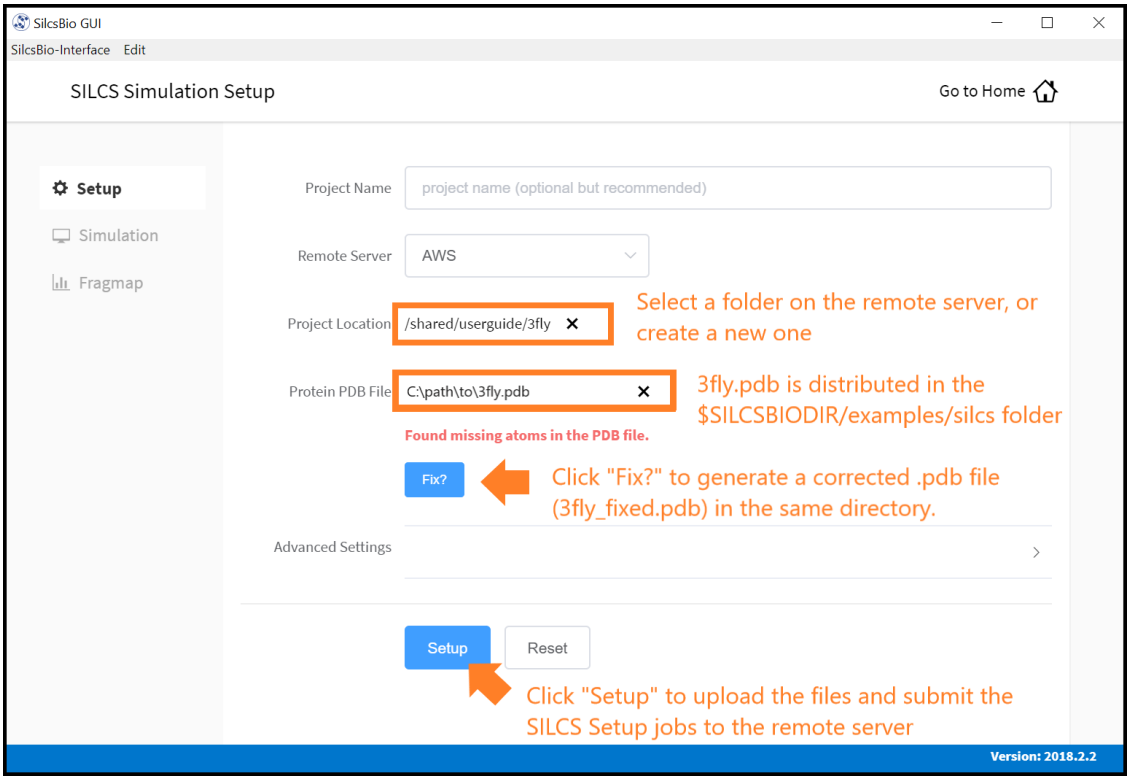


Once all information is entered, click the "Save" button. The GUI will save the information and confirm that you have a working connection to the server.

2.2 SILCS simulation

To begin a new SILCS project, follow these steps:

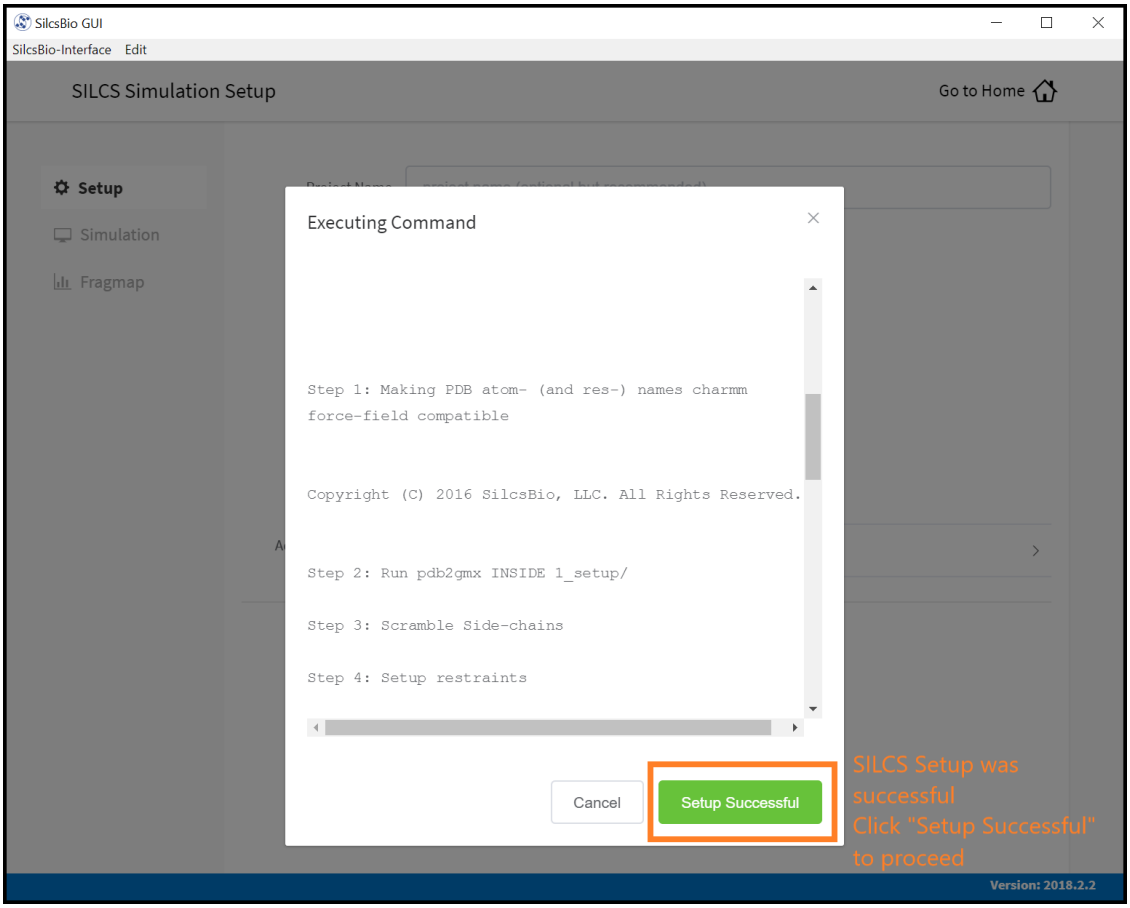
1. Select *New SILCS project* from the Home page.
2. Enter a project name, select your remote server, and select a project location file folder on the remote server. Typical SILCS simulations produce output files in excess of 100 GB, so please select a project location file folder with appropriate storage capacity.



Next, select a 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.

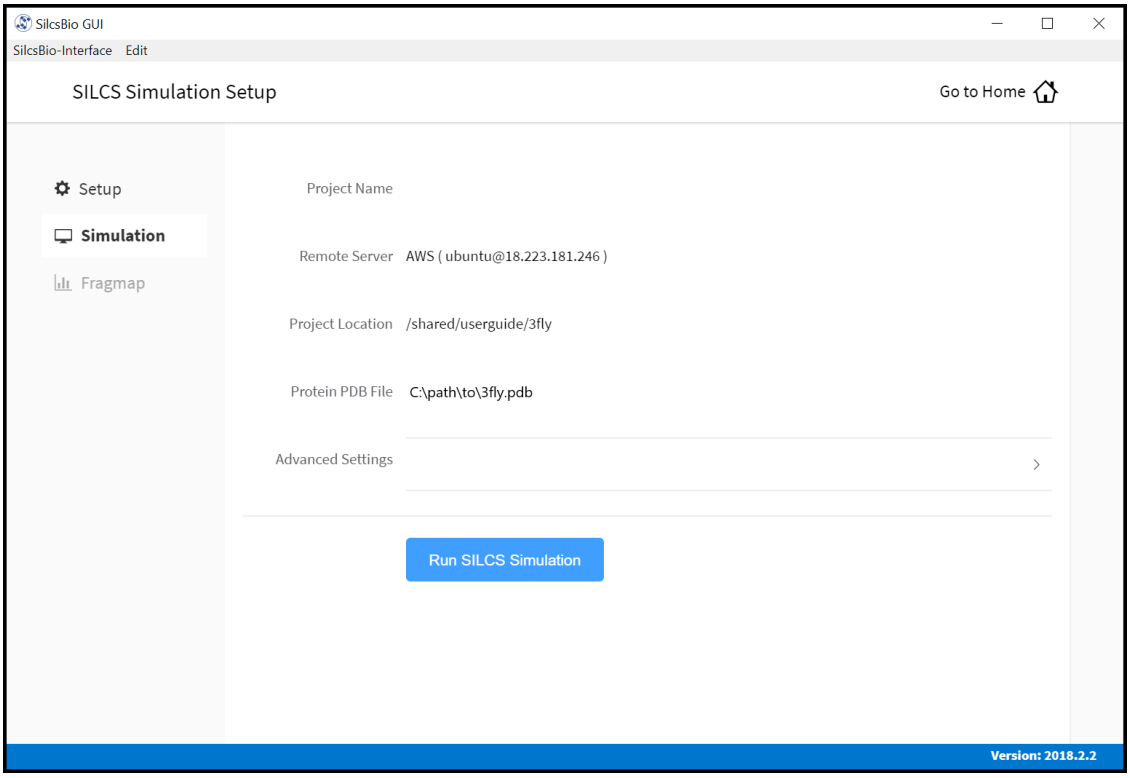
If the GUI detects missing non-hydrogen atoms, non-standard residue names, or non-contiguous residue numbering, it will inform the user and provide a button labeled “Fix?”. If this button is clicked, a new PDB file with these problems fixed and with `_fixed` added to the base name will be created and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

3. 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 SILCS GCMC/MD setup process.



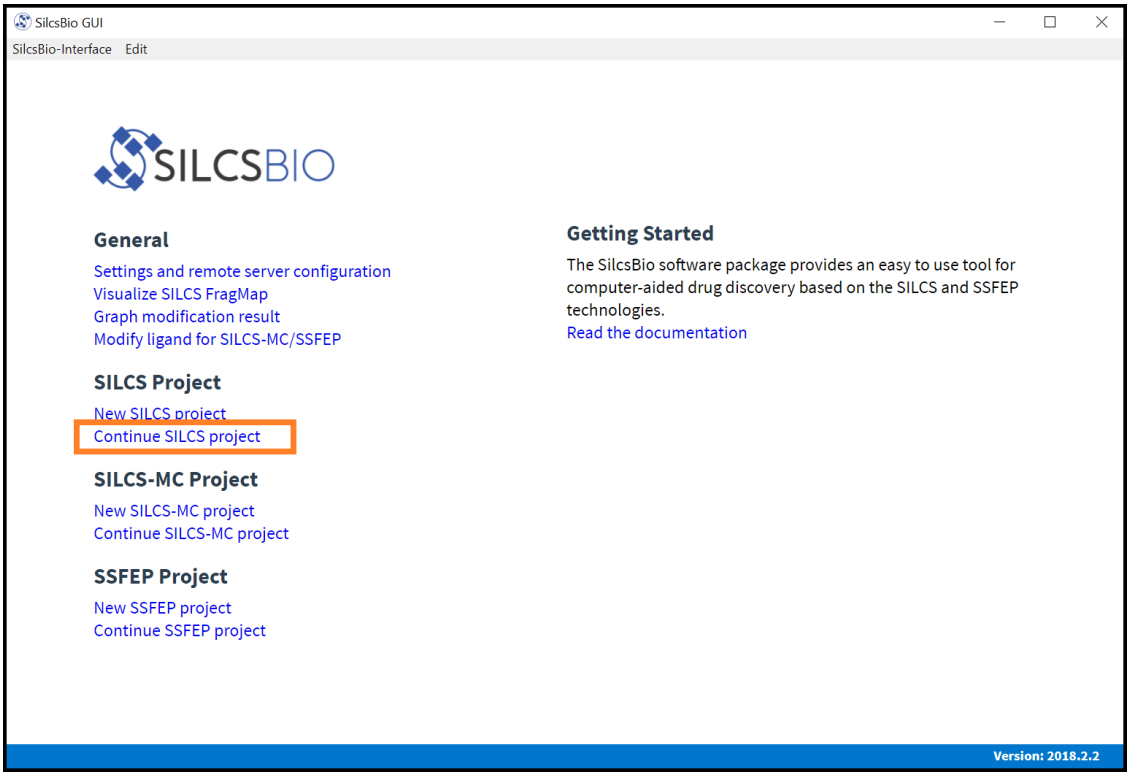
During setup, the program automatically performs several steps: building the topology of the simulation system, creating metal-protein bonds if metal ions are found, rotating side chain orientations to enhance sampling, and putting probe molecules around the protein. To complete the entire process may 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.

4. 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.



10 compute jobs will be submitted to the queueing system 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, "R" for running, and "E" for finished. 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.

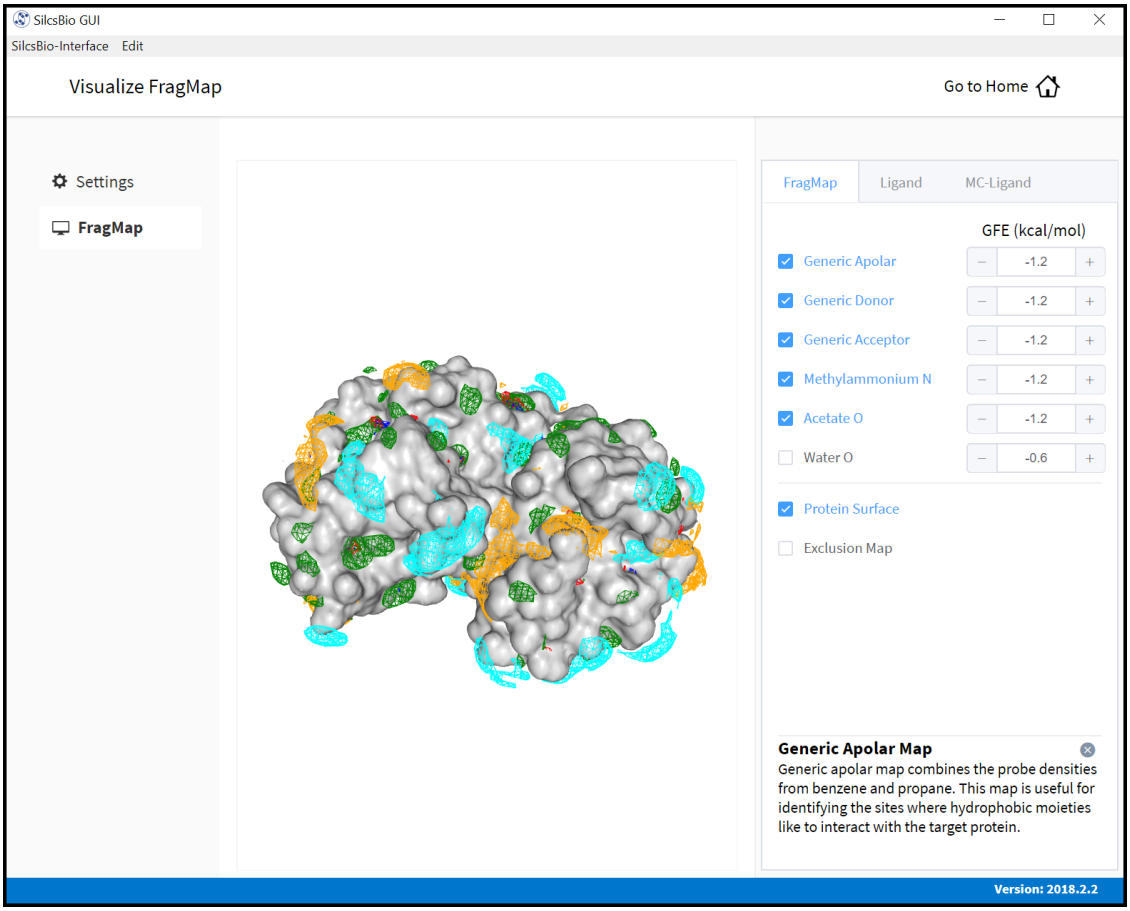
5. To see a full listing of all of your projects, select *Continue SILCS project* from the Home page. This will show the complete list of all SILCS projects you have set up on the local machine where you are currently running the SilcsBio GUI, as well as the status of each project.



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.

6. Once your SILCS project compute jobs are finished, the GUI can be used to create FragMaps and visualize them. Confirm project location and the input protein PDB file are correct. Select the reference PDB file to be used to generate FragMaps. 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.

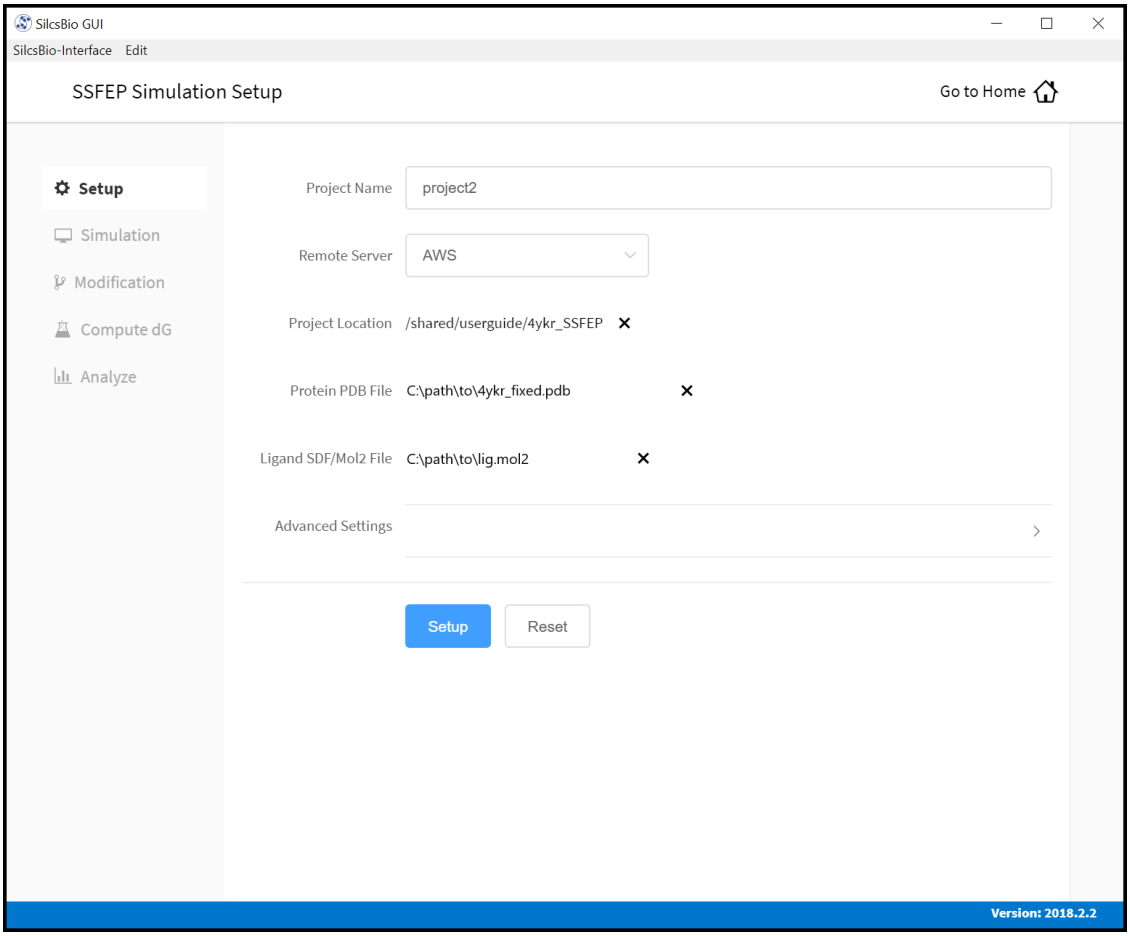
Tip: If you plan to compare FragMaps from two different protein structures, you will want to generate them with the same orientation. In that case, pre-align your input structures with each other and use these aligned coordinates as your Reference PDB Files.



For additional details, please see *Site Identification by Ligand Competitive Saturation (SILCS)*.

2.3 SSFEP simulation

1. Select *New SSFEP project* from the Home page.
2. Enter a project name, select your remote server, and select a project location file folder on the remote server. Typical SSFEP simulations produce output files in excess of 20 GB, so please select a project location file folder with appropriate storage capacity.



Next, select a protein PDB file and a ligand file. The ligand should be aligned to the binding pocket in the accompanying protein PDB file. We 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. 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.

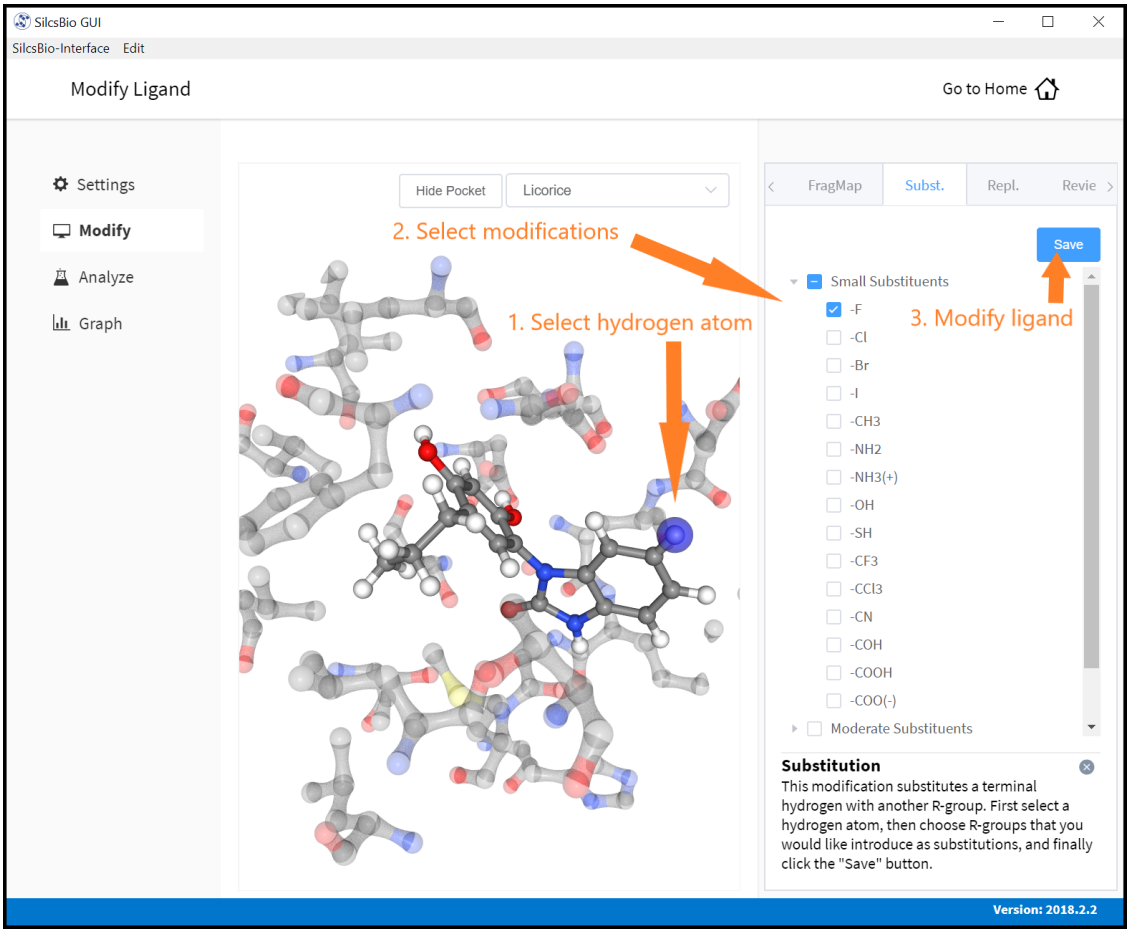
   During setup, the program automatically performs several steps: building the topology of the simulation system, creating metal-protein bonds if metal ions are found, rotating side chain orientations to enhance sampling, and putting probe molecules around the protein. To complete the entire process may 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 proceed.

4. Your SSFEP simulation can now be started by clicking the "Run MD 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 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. The rapid $\Delta\Delta G$ calculation part relies on the MD results and 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, "R" for running, and "E" for finished.

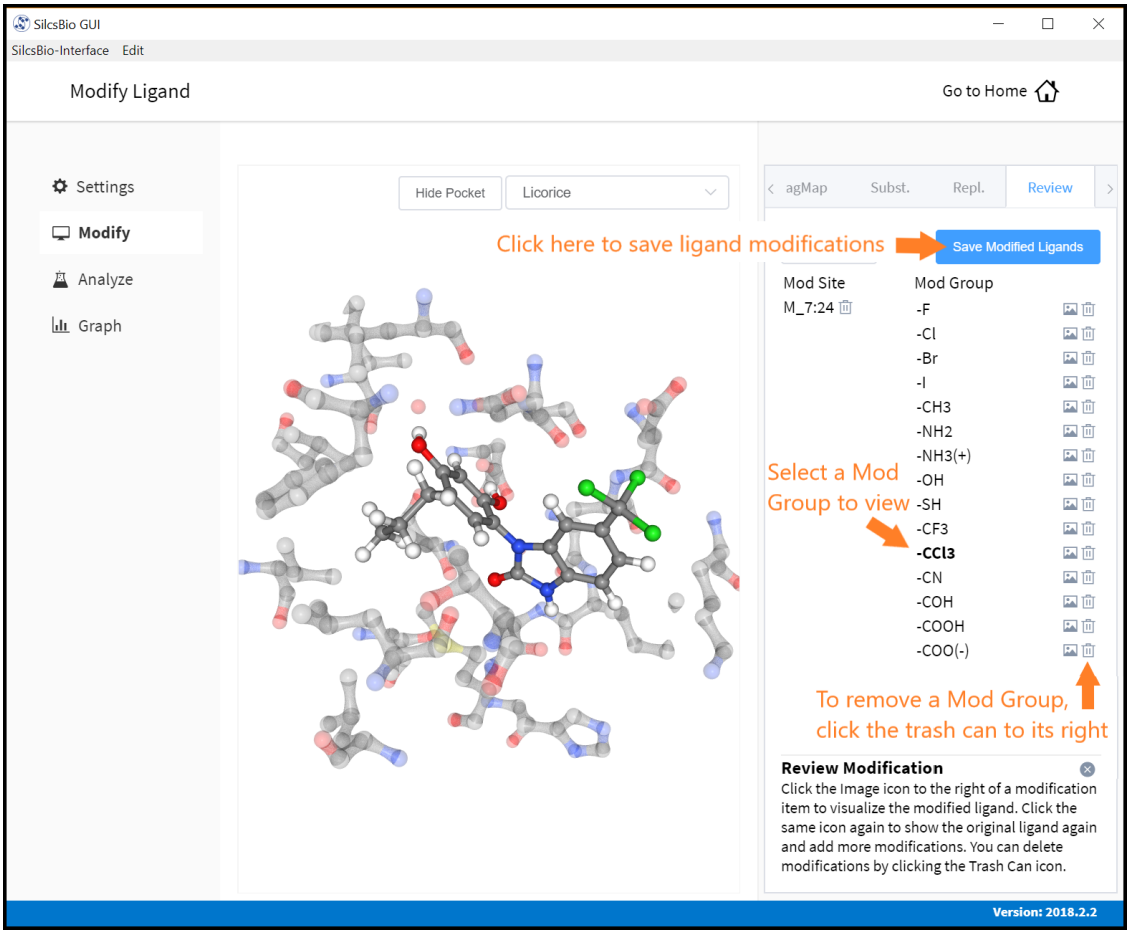
5. While your jobs are running, you can prepare your ligand modifications with the "Add Ligand Modifications" button.

   This will show the parent ligand in the binding pocket, and the binding pocket atoms may be visualized in licorice or surface representation. 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 "Add" 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 *Single Step Free Energy Perturbation (SSFEP)*.

6. 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**.

8. Once MD simulation is finished, the GUI will analyze your list of modifications and create a chart. 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, please see *Single Step Free Energy Perturbation (SSFEP)*.

CHAPTER

THREE

QUICKSTART GUIDE: COMMAND LINE INTERFACE

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

3.1 SILCS simulation

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 the progress of the jobs, use the `tail -n1 2a_run_gcmd/*/.progress` command. You will see two numbers for each job. The first number is the current cycle number and the second number is the total number of cycles required to complete the simulation.

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

```
${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, use the following command to create the GFE FragMaps:

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

This command will create a `silcs_fragmap_<Protein PDB>` folder, which contains final FragMap and VMD/PyMol scripts for easy visualization:

```
pymol view_maps.pml # PyMol
vmd -e view_maps.vmd # VMD
```

Additional files that are created include those needed for subsequent SILCS-MC docking and the file `overlap_coefficient.dat`. The overlap coefficients in this file are a measure of convergence for SILCS simulations. If the coefficients are less than 0.7, you may consider extending the SILCS simulations.

For additional details, please see *Site Identification by Ligand Competitive Saturation (SILCS)*.

3.2 SSFEP simulation

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 the progress of the jobs, use the `tail -n1 2_run_md/*/.progress` command. You will see two numbers for each job. The first number is the current cycle number and the second number is the total number of cycles required to complete the simulation. SSFEP MD simulations may take 6-12 hours to complete depending on the hardware and system size.

4. 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 < 0 indicate a modification predicted to be favorable.

For more details, please see *Single Step Free Energy Perturbation (SSFEP)*.

CHAPTER

FOUR

SILCSBIO SOFTWARE INSTALLATION

4.1 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 80-90 hours of walltime when run in parallel on ten compute nodes, each equipped with 8 3-GHz CPU cores. The software can take advantage of GPU acceleration: the addition of a single NVIDIA GeForce GTX 980 GPU to each node will reduce the walltime to 24-48 hours. 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, subsequent evaluation of relative ligand affinities takes on the order of 10 minutes on a single CPU core, allowing for modifications to be rapidly evaluated on ordinary laptop or desktop computers. For customers without ready access to an appropriate computing cluster, SilcsBio is able to perform these computations as a service and supply data to the customer for subsequent in-house analysis. For this service, in the case of SILCS, SilcsBio requires only the structure of the target, and, in the case of SSFEP, only the structure of the protein-parent ligand complex. Depending on the 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. Alternatively, SilcsBio can assist customers with setting up their own virtual cluster using Amazon Web Services. Please contact info@silcsbio.com for additional information.

4.2 Software requirement

SILCS FragMap generation and SSFEP calculations use the MD simulation package GROMACS. Although the SilcsBIo software is compatible wth GROMACS version 5.1.0 and later, we recommend GROMACS version 2018.3. The package can be obtained at http://manual.gromacs.org/documentation/2018.3/download.html and must be installed in order to use the SilcsBio software package.

Below is a recommended sequence of commands for general installation of GROMACS (with GPU acceleration):

```
cd <GROMACS source directory>
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=on \
  -DREGRESSIONTEST_DOWNLOAD=on \
  -DGMX_GPU=on \
  -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 command:

```
cd <GROMACS source directory>
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=on \
  -DREGRESSIONTEST_DOWNLOAD=on \
  -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 detail.

4.3 Installation

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

```
silcsbio/
  data/
  examples/
  lib/
  programs/
  silcs/
  silcs-mc/
```

(continues on next page)

(continued from previous page)

```
  silcs-memb/
  ssfep/
  ssfep-memb/
  templates/
  test/
  utils/
  VERSION
```

The `silcsbio` folder contains software for running *SILCS* and *SSFEP* simulations. The `program`, `silcs`, and `ssfep` folders contain executable code, and the `templates` folder contains templates for job submission and input scripts. Some template files will need to be edited with information for your queuing system.

Uncompress and place the folders in an appropriate location. If you are a system administrator, place the folder where it can be accessible by other users, such as `/opt/silcsbio/`. If you are a single user, you may place the folder in your home directory.

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

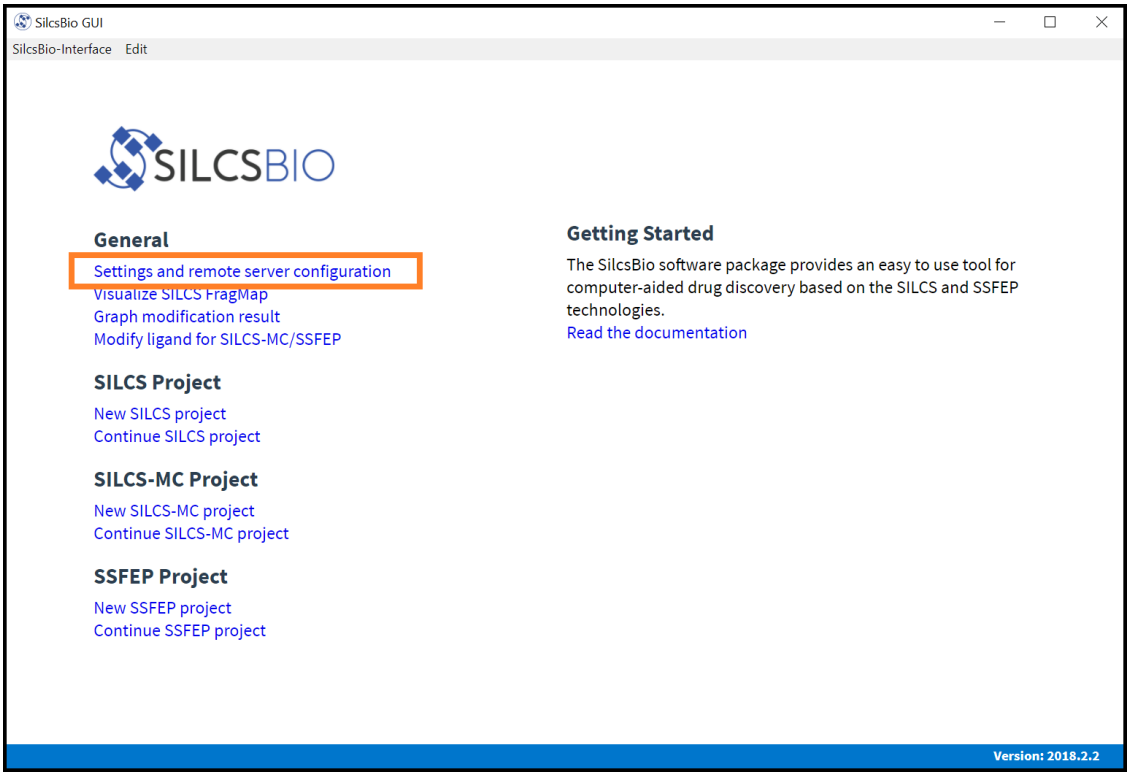
```
# bash
export GMXDIR=<gromacs/bin>
export SILCSBIODIR=<silcsbio>
```

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

4.4 Installing SilcsBio Graphical User Interface

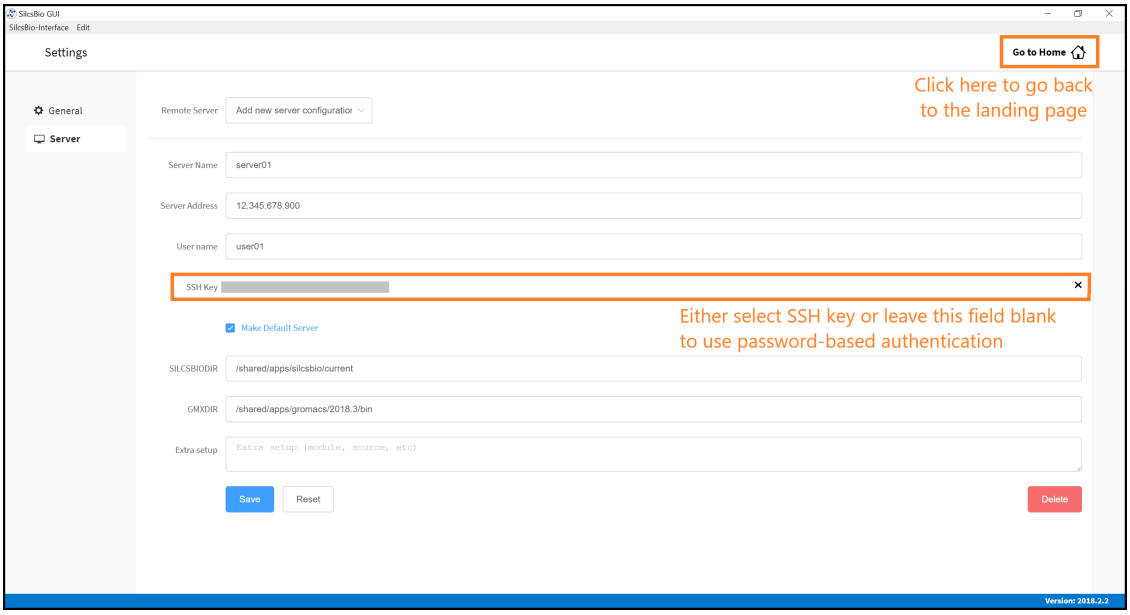
The SilcsBio Graphical User Interface (GUI) enables running SILCS and SSFEP simulations and analyzing results through a GUI instead of the command line. Supported platforms are macOS and Windows. Please download and install the software on your local desktop or laptop computer.

In addition to standalone features such as FragMap visualization and introducing ligand modifications, the SilcsBio GUI has the power to set up, launch, manage, and analyze compute-intensive SILCS and SSFEP simulations. To enable this functionality requires a simple configuration step to allow the GUI to communicate with your remote computing cluster. When you launch the GUI, select *Settings and remote server configuration*.



Within the "Settings" page, select *Server* menu from the left-hand column. In the main panel, enter the "Server Name", "Server Address" (IP address), "User name" for server login, and "SSH key". If you do not have an SSH key for the server, leave it blank; the GUI will ask you the password to the remote server instead of using passwordless key login. Select the "Make Default Server" checkbox if you would like to set this server as your default server, causing this server to be selected as the default in other parts of the interface.

Enter `SILCSBIODIR` and `GMXDIR` information that matches the values selected in the previous section.



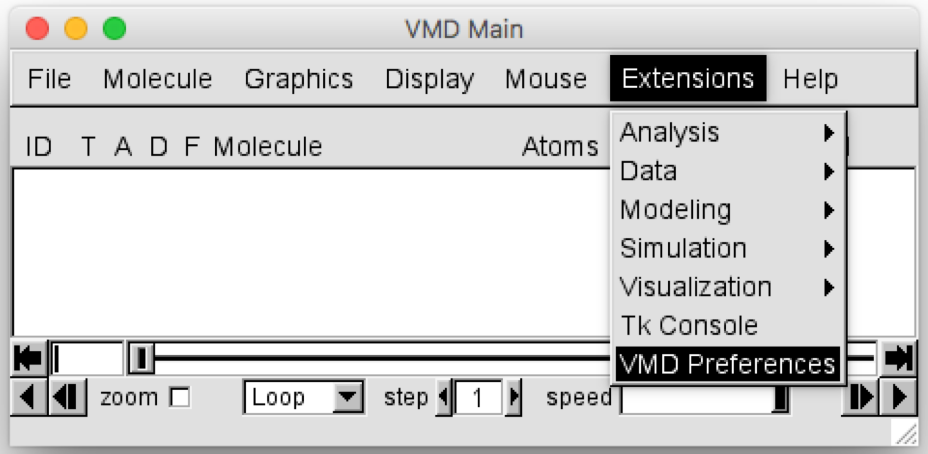
Once you have completed entering all values, click the "Save" button. The interface will test the connection and store the server information for your future use. Please contact support@silcsbio.com if you need help with this process.

4.5 Installing visualization plugins

Note: These plugins are used for visualization of FragMaps. If you are only interested in SSFEP, you may skip this section.

4.5.1 VMD plugin installation

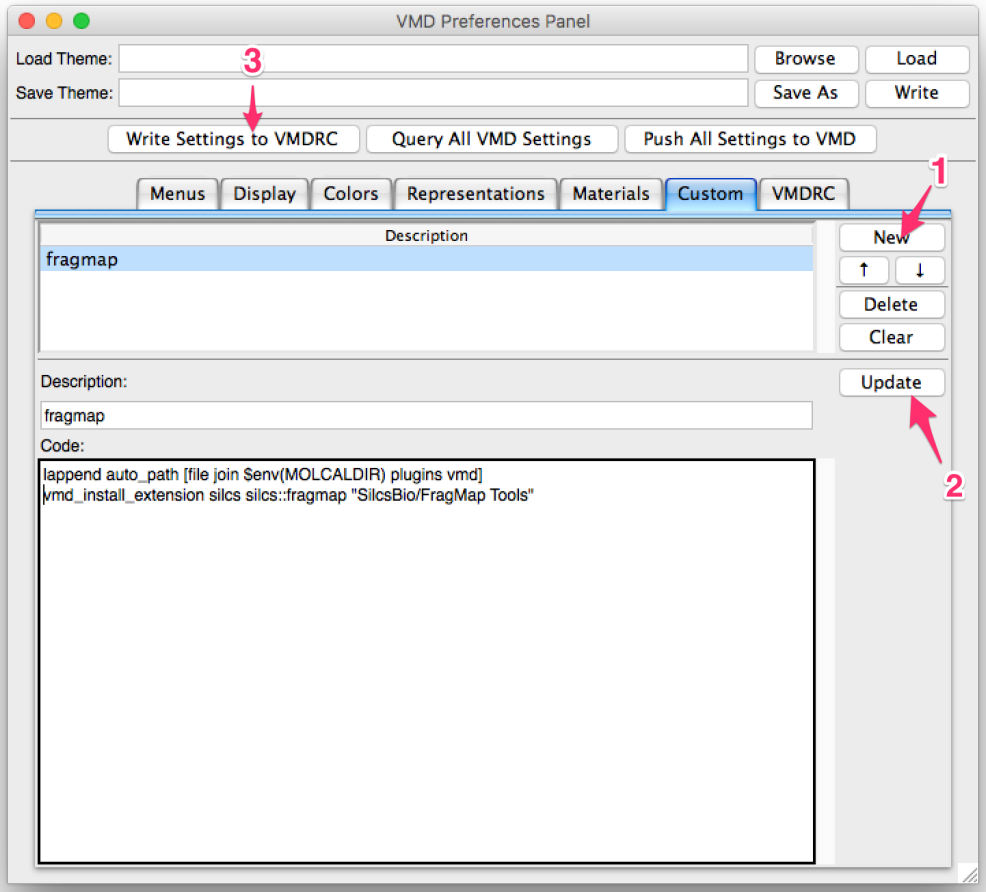
For VMD versions later than 1.9, the plugin can be installed using the VMD Preference menu, which can be found in the "VMD Main" window using the menu selection *Extensions* → *VMD Preferences*:



In the preference window, select the "Custom" tab and press the "New" button. Enter the following lines in the "Code" section:

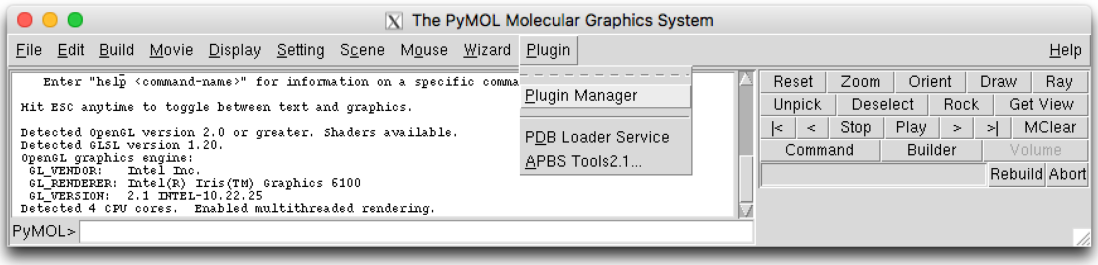
```
lappend auto_path [file join $env(SILCSBIODIR) utils plugins vmd]
vmd_install_extension silcs silcs::fragmap "SilcsBio/FragMap Tools"
```

If you have not set the environment variable `SILCSBIODIR`, replace `$env(SILCSBIODIR)` with the full path to the `silcsbio` folder. Enter an appropriate name for the plugin under "Description:", then press the "Update" button. Finally, press the "Write Settings to VMDRC" button and then restart VMD to confirm the plugin installation.



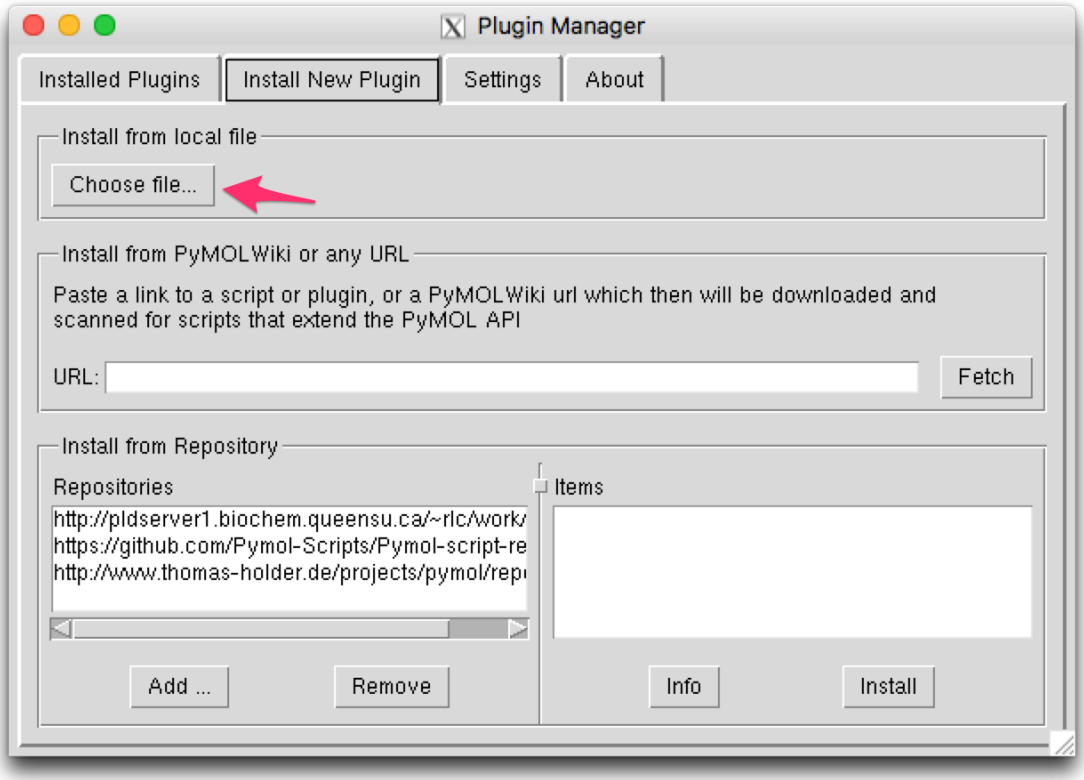
4.5.2 PyMOL plugin installation

The PyMOL plugin can be installed using the plugin manager, which can be found using the PyMol menu selection *Plugin* → *Plugin Manager*:



This will open a plugin manager window. Select the "Install New Plugin" tab. Press the "Choose file…" button in the "Install from local file" section, then choose `silcsbio.2019.1/utils/`

`plugins/pymol/fragmap_tools.py`. This will install the PyMOL plugin. Restart PyMOL to confirm the installation.



CHAPTER

FIVE

FREQUENTLY ASKED QUESTIONS

1. 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 submission 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 nproc=1
```

If the script failed to run at the `1_setup-silcs_boxes` step, the software is not correctly installed, whereas if the script failed to run at the `2a_run_gcmd` step, the job handling script needs 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 the job submission 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 the script failed to run at the `1_setup_ssfep` step, the software is not correctly installed, whereas if the script failed to run at 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 the job submission script requires editing. Please contact support@silcsbio.com if you need assistance.

2. I don't have a cluster but I have a GPU workstation. What can I do?

We recommend the use of compute clusters over workstations, due to the computational resources required to perform simulations (especially in the case of SILCS). Typical timing information given in the introduction section is based on using 10 compute nodes all at the same time. However, as workstations with powerful GPUs are becoming more common, questions about whether a single GPU workstation can be used as a compute node are increasing.

For those users with a GPU workstation, we offer a simple queue system. This queue system is intended only for use on a single compute node and is designed to mimic a typical queue system, but lacks more sophisticated features like resource control or multiple compute node support.

This queue system can be found in the download page. Download and place the files in a folder that is in your `PATH`. If you have administrtive privileges on the workstation, place the files in a widely-accessible directory.

Once the files are placed, start up the scheduler by using the following command:

```
scheduler --workers=<number of workers>
```

Set the number of workers as the number of parallel jobs that can be run at the same time. For example, if the workstation has 2 GPUs, set the number of workers as 2.

This will require the user to adjust the `nproc` variable appropriately while using SILCS or SSFEP job handling scripts. Typically, `nproc = (total number of CPUs) / (number of workers)`.

For example, if the workstation has 16 CPUs and 2 GPUs, then you may want to run

two jobs at the same time, with each using 8 CPUs and one GPU. In this case, set the number of workers to 2 and the `nproc` variable to 8.

3. 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 edit the 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"
```

4. GROMACS on the head node does not run because the head node and compute node have different operating systems and GROMACS was compiled on the compute node.

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"
```

5. I got "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 culprit 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 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 the folder in `LD_LIBRARY_PATH`.

If the library is not available on the current machine, we recommend following FAQ #4 to compile GROMACS and `mdrun` separately.

CHAPTER

SIX

SITE IDENTIFICATION BY LIGAND COMPETITIVE SATURATION (SILCS)

6.1 Background

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 protein *[5]*. The SILCS method yields functional group free energy maps, or FragMaps, which are precomputed and rapidly used in a variety of ways to facilitate ligand design. FragMaps are generated by molecular dynamics (MD) simulations that include protein flexibility and explicit solvent/solute representation, thus providing an accurate, detailed, and comprehensive collection of information that can be used qualitatively in database screening, fragment-based drug design and lead optimization.

In addition, 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.

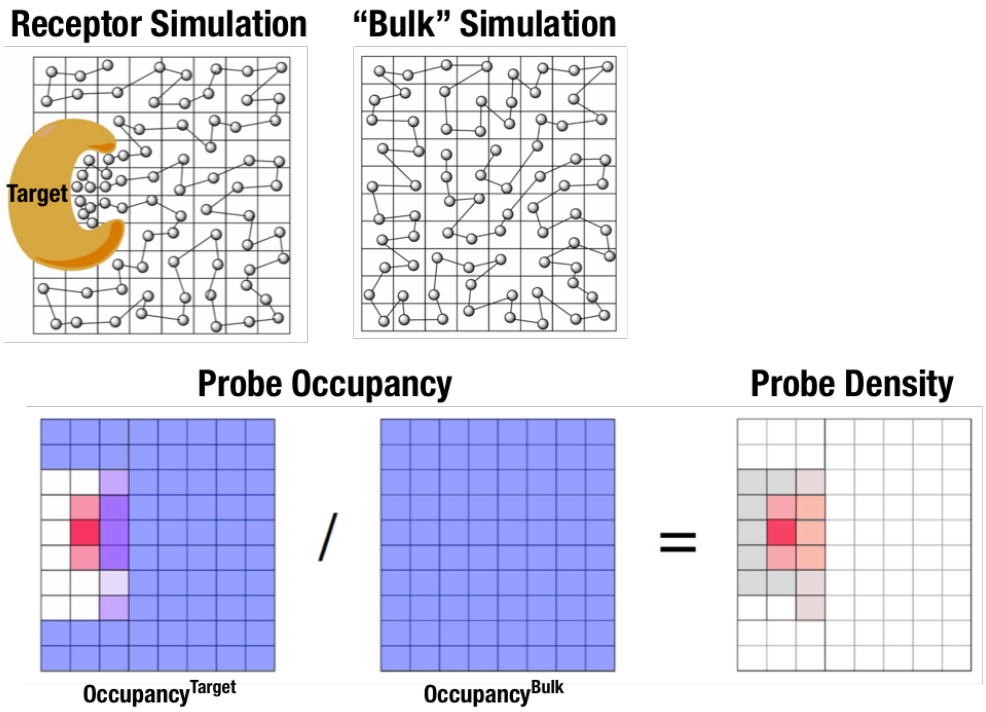


Fig. 6.1: Illustration of fragment density maps.

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

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

By generating FragMaps using various small solutes representing different functional groups including benzene, propane, methanol, imidazole, formamide, acetaldehyde, methylammonium, acetate, and water, 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.

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 by water or probe molecules during the MD simulations. 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 *[8]*. Thus, the SILCS technology is especially well-suited for targeting the deep and inaccessible binding sites found in targets like GPCRs and nuclear receptors.

Once a set of ligand-independent SILCS FragMaps are produced they may be rapidly used for various purposes that quantitatively 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
- Ligand optimization can also be performed via rapid free energy perturbations of small functional group transformations via the SSFEP technology (please see *Single Step Free Energy Perturbation (SSFEP)*).

SILCS generates 3D maps of interaction patterns of functional group with your target molecule, called FragMaps. This unique approach simultaneously uses multiple small solutes with various functional groups in explicit solvent MD simulations that include target flexibility to yield 3D FragMaps that represent 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 computationally demanding precomputation of the FragMaps.

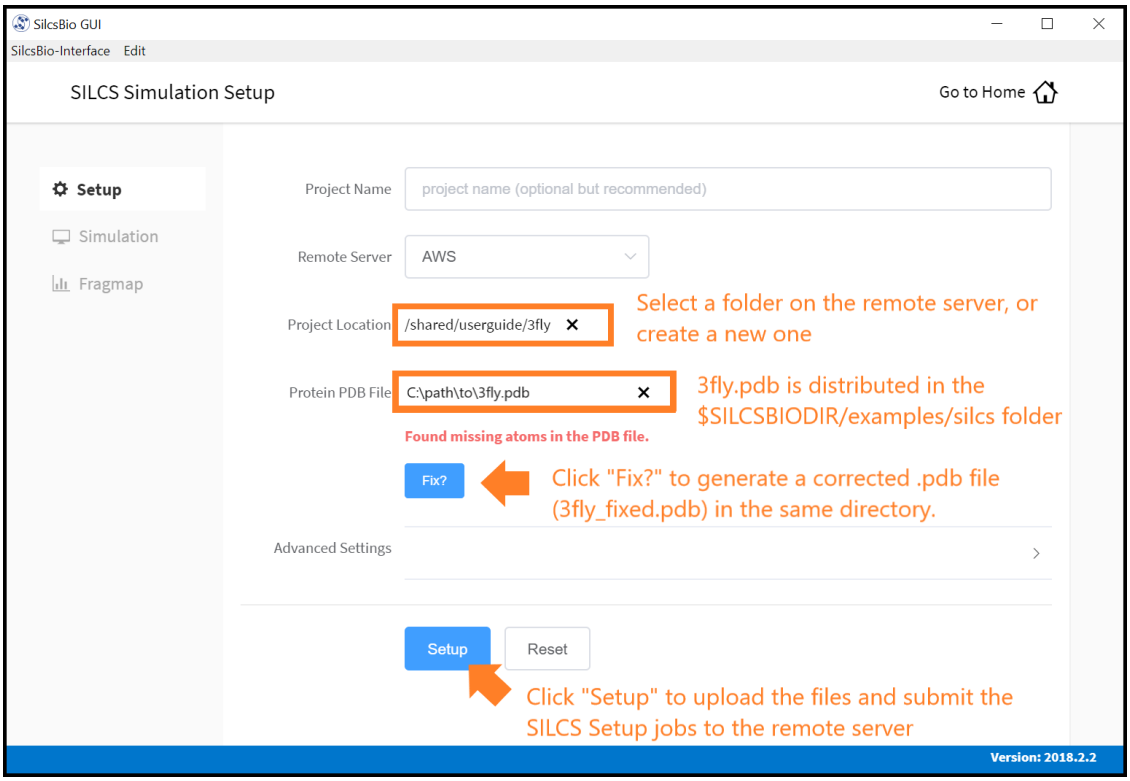
This chapter goes over the workflow of generating and visualizing FragMaps.

6.2 Running SILCS simulations from the SilcsBio GUI

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 inquire if assistance with protein preparation is required.

6.2.1 SILCS simulation setup

Select *New SILCS project* from the landing page. This will take you to the new project setup page.

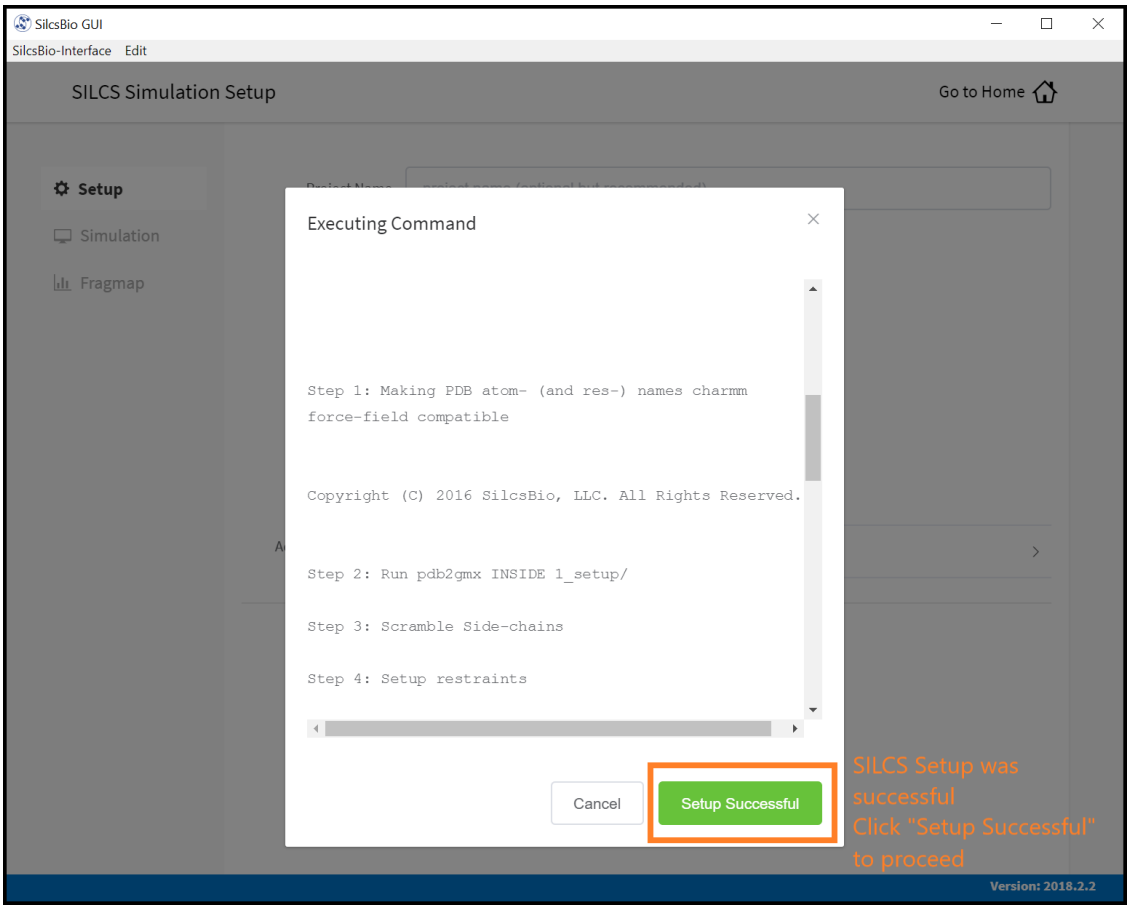


Enter a project name, select a remote server, and select a project location on the remote server. The project location is where the SILCS simulation will be performed. A typical SILCS simulation can produce output totaling in excess of 100 GB, so please select a folder with appropriate free space.

Next, select a protein PDB file. We recommend clean up PDB before using in SILCS simulation, such as only keeping the chains that are necessary for the simulation, removing any unnecessary ligands, renaming non-standard residues, and filling in the missing atomic positions.

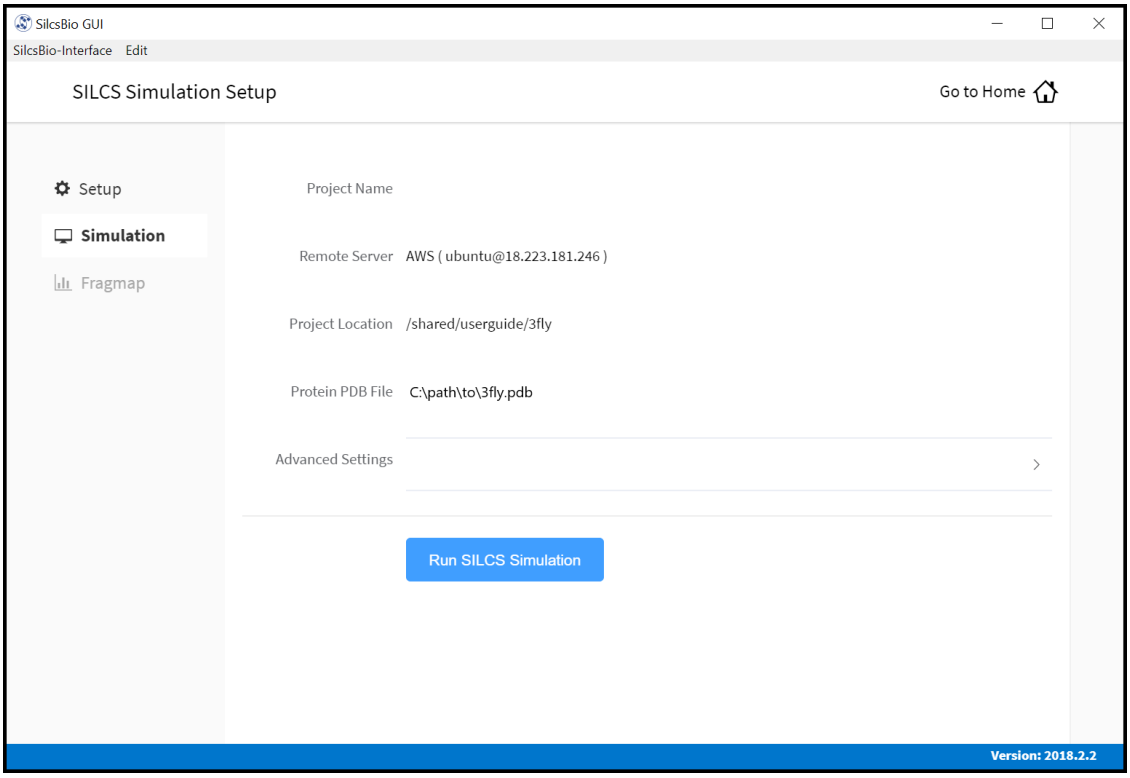
If the interface detects missing heavy atoms, non-standard residue name, or non-contiguous residue numbering, it will inform the user and provide a button labeled as “Fix?”. If this button is clicked, a new PDB file with “_fixed” suffix will be created and used in the SILCS simulation.

Once all the information is entered correctly, press the “Setup” button at the bottom of the page. The interface will contact the remote server and perform the SILCS GCMC/MD setup process.



This step 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. This process may take 5-10 minutes depending on system size.

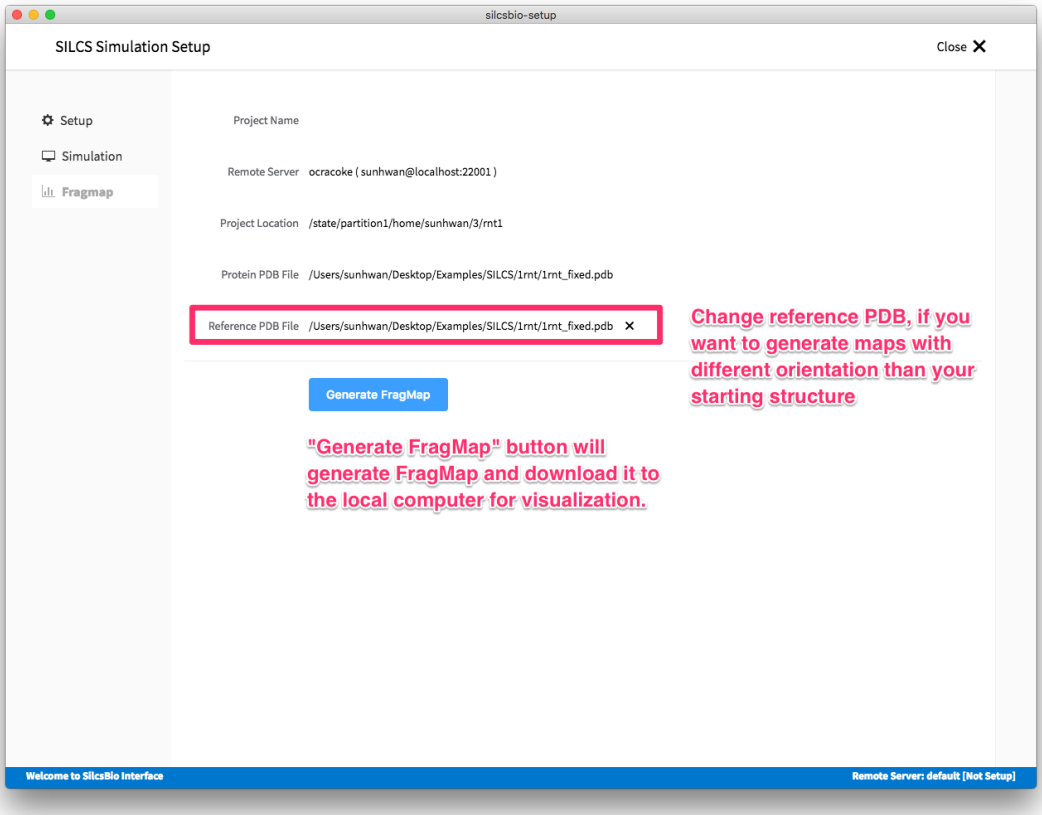
After confirming the setup is finished, please make sure information entered are correct and the remote project location has an ample storage space. When it is ready, SILCS GCMC/MD simulation can be started by clicking the “Run SILCS Simulation” button.



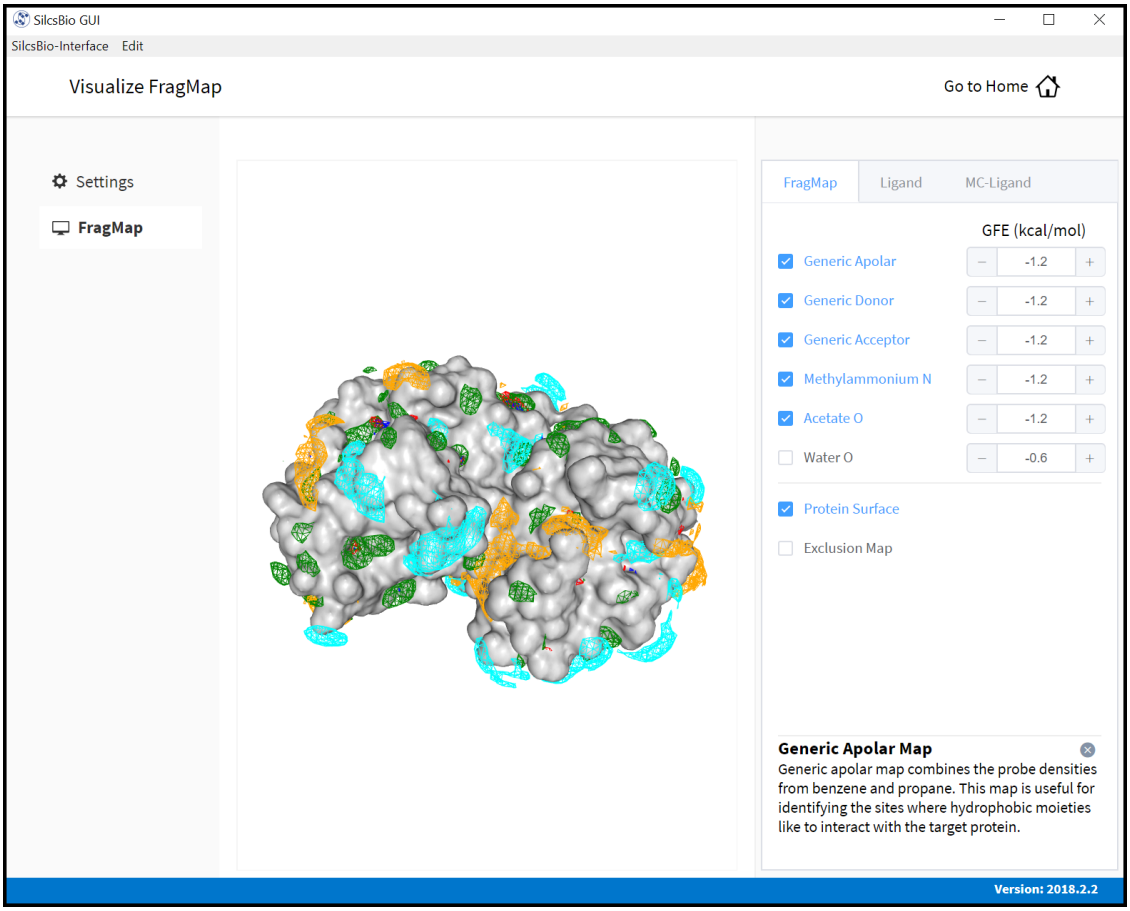
This will submit 10 jobs to the queue system. The progress of the jobs will be displayed on the same page. The status of the job is also shown next to the progress bar; "Q" in the queue, "R" for running, and "E" for finished.

6.2.2 FragMaps generation

Once the simulation is finished, the GUI can be used to create and visualize FragMaps.



In case you want to compare FragMaps from two different protein structures, you should pre-align the input structures and use the aligned structure as the Reference PDB File to create FragMaps having the desired orientation.



6.3 Running SILCS simulations from the command line interface

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 inquire if assistance with protein preparation is required.

6.3.1 SILCS simulation setup

```
${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.

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.

Once the simulations are complete, the `2a_run_gcmd/[1-10]` directories will contain `*.prod.125.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.

6.3.2 FragMap generation

Once your simulations are done, generate the FragMaps using the following script.

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

This script will submit 10 single-core jobs. These will build occupancy maps spanning the simulation box for selected 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 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>
```

GFE FragMaps will be placed in the `silcs_fragmap_<protein PDB>/maps` directory. PyMOL and VMD scripts to loads these file will be place in the `silcs_fragmap_<protein PDB>` directory.

```
<prot name>.benc.gfe.map     Aromatic map
<prot name>.prpc.gfe.map     Aliphatic map
<prot name>.aceo.gfe.map     Charged acceptor map
<prot name>.mamn.gfe.map     Charged donor map
<prot name>.forn.gfe.map     Polar nitrogen donor map (Formamide)
<prot name>.foro.gfe.map     Polar oxygen acceptor map (Formamide)
<prot name>.imin.gfe.map     Polar nitrogen acceptor map (Imidazole)
<prot name>.iminh.gfe.map    Polar nitrogen donor map (Imidazole)
<prot name>.aalo.gfe.map     Polar oxygen acceptor map (Acetaldehyde)
<prot name>.meoo.gfe.map     Polar oxygen acceptor map (Methanol)
<prot name>.apolar.gfe.map   Generic Apolar maps
<prot name>.hbdon.gfe.map    Generic donor maps
<prot name>.hbacc.gfe.map    Generic acceptor maps
<prot name>.excl.map         Exclusion map
<prot name>.ncla.map         Zero map
```

6.3.3 Cleanup

Raw trajectory files are large. To reduce file sizes, hydrogen atom positions can be deleted with the following:

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

6.3.4 SILCS simulation setup with GPCR targets

SilcsBio provides a command line utility to embed transmembrane proteins, such as GPCRs, in a bilayer of 9:1 POPC/Cholesterol 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 the 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}/silcs-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}/silcs-memb/1a_fit_protein_in_bilayer prot=<Protein PDB>␣
↪bilayer_x_size=<X dimension> bilayer_y_size=<Y dimension>
```

The resulting protein/bilayer system will be output in a PDB file with suffix `_popc_chol`. It is important to use a 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 that has been built using other software tools.

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

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

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

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

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

FragMap generation from the GCMC/MD simulation 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.

6.4 Visualizing FragMaps

GFE FragMaps are created in the directory `2b_gen_maps`. This folder contains the specific and generic FragMaps listed previously. FragMaps are defined using on non-hydrogen atoms. For certain solute types there are multiple FragMaps (imidazole, formamide). FragMaps for water oxygens (tipo) are included to identify water molecules that are 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 visualization.

The easiest way to visualize FragMaps is with the SilcsBio GUI. SilcsBio also provide plugins for the visualization softwares VMD and PyMOL:

6.4.1 FragMaps in VMD

Please ensure the SilcsBio VMD plugin is installed before continuing (see *VMD plugin installation*).

First, load the protein structure used for FragMap generation. Once the protein structure is loaded, select *Extensions* → *SilcsBio* → *FragMap Tools* to activate the plugin window.

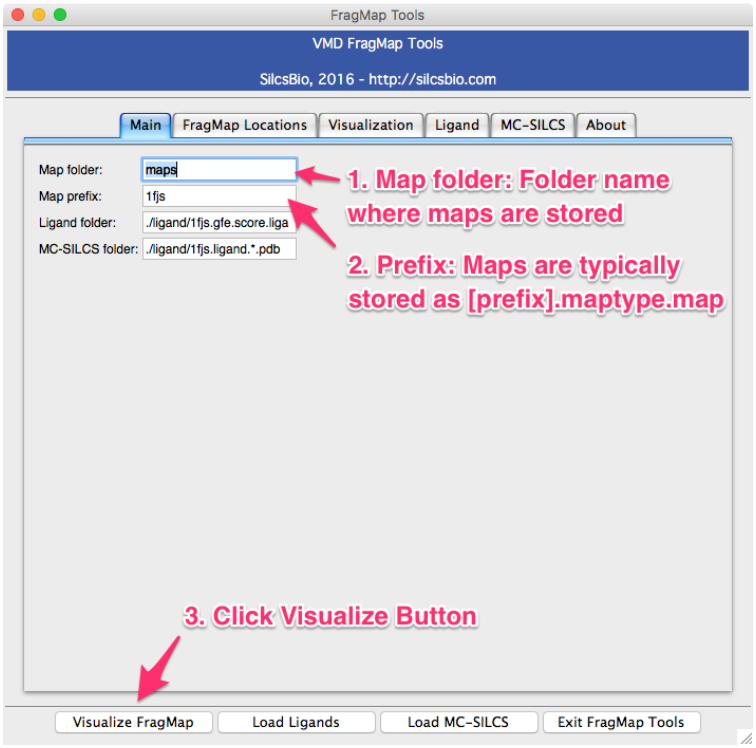
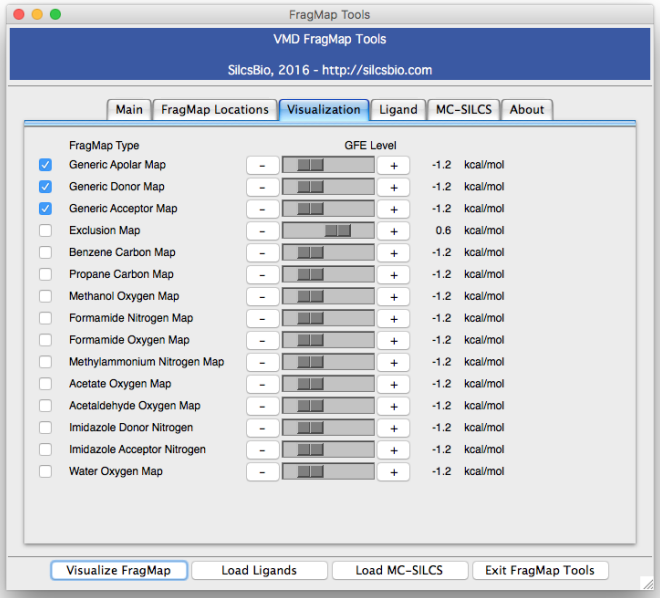


Fig. 6.2: VMD FragMap Plugin

"Map folder" is the name of the folder that contains FragMap files, and is typically "2b_gen_maps". "Map prefix" is the protein PDB name used when setting up and performing the SILCS simulations. If a protein PDB file was loaded in to VMD prior to opening the plugin window, the name of the PDB file will be taken as a default for this parameter. Once these two parameters are confirmed, click the "Visualize FragMap" button at the bottom of the window.

At this point, FragMaps will have been loaded in to the VMD Main window and a new tab will have appeared in the plugin window. This new tab allows quick enabling/disabling of FragMap display using checkboxes on the left hand side of the window. By default, GFE FragMaps are displayed at a contour level of -1.2 kcal/mol. This level can be changed by using either the +/- buttons or the slider.



6.4.2 FragMaps in PyMOL

Please ensure the SilcsBio PyMol plugin is installed before continuing (see *PyMOL plugin installation*).

First, load the protein structure used for FragMap generation. Once the protein structure is loaded, select *Plug* → *FragMap Tools* to activate the plugin window.

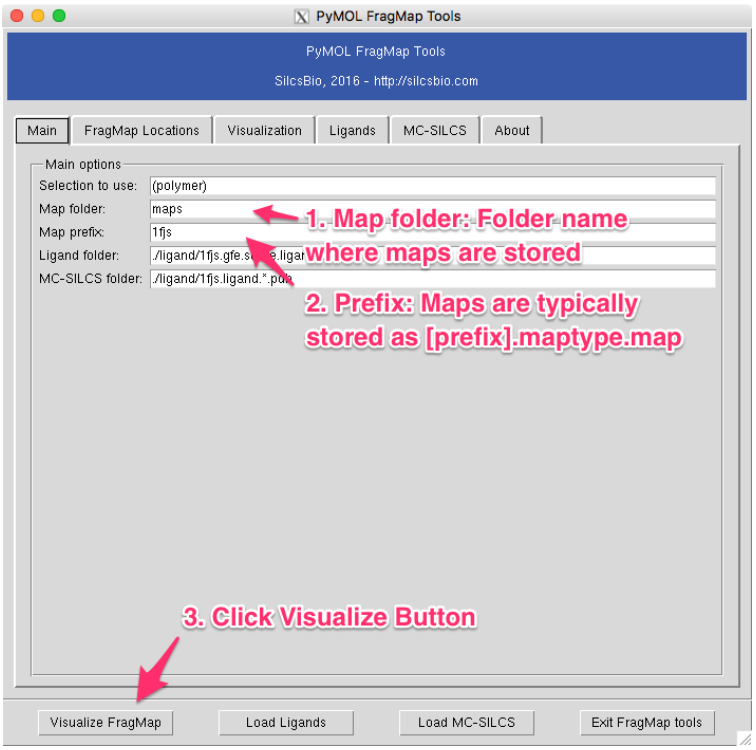
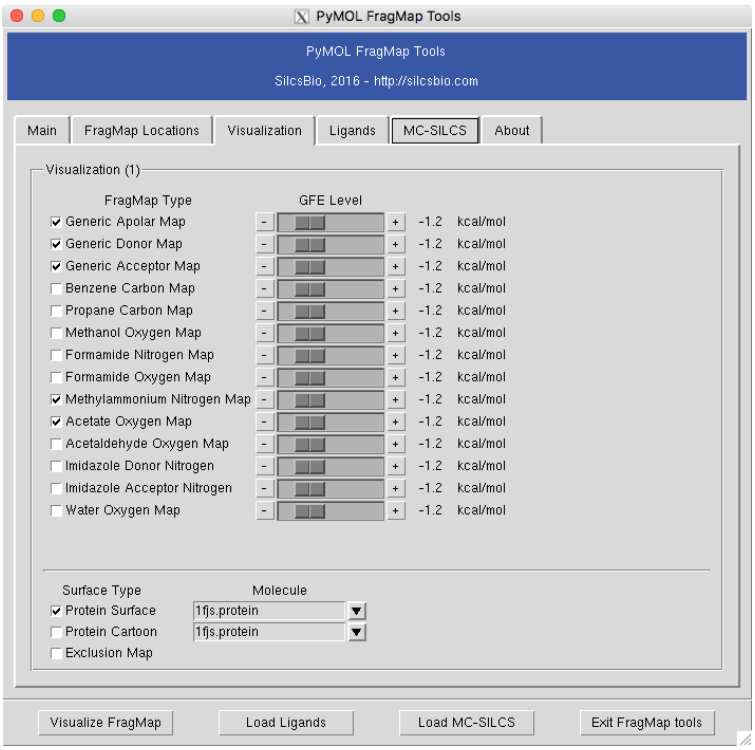


Fig. 6.3: PyMOL FragMap Plugin

"Map folder" is the name of the folder that contains FragMap files, and is typically "2b_gen_maps". "Map prefix" is the protein PDB name used when setting up and performing the SILCS simulations. If a protein PDB file was loaded in to PyMol prior to opening the plugin window, the name of the PDB file will be taken as a default for this parameter. Once these two parameters are confirmed, click the "Visualize FragMap" button at the bottom of the window.

At this point, FragMaps will have been loaded in to the Pymol main window and a new tab will have appeared in the plugin window. This new tab allows quick enabling/disabling of FragMap display using checkboxes on the left hand side of the window. By default, GFE FragMaps are displayed at a contour level of -1.2 kcal/mol (with the exception of the Exclusion Map). This level can be changed by using either the +/- buttons or the slider.


PyMOL FragMap Tools
PyMOL FragMap Tools
SilcsBio, 2016 - http://silcsbio.com
Main
FragMap Locations
Visualization
Ligands
MC-SILCS
About
Visualization (1)
FragMap Type
GFE Level
Generic Apolar Map
Generic Donor Map
Generic Acceptor Map
Benzene Carbon Map
Propane Carbon Map
Methanol Oxygen Map
Formamide Nitrogen Map
Formamide Oxygen Map
Methylammonium Nitrogen Map
Acetate Oxygen Map
Acetaldehyde Oxygen Map
Imidazole Donor Nitrogen
Imidazole Acceptor Nitrogen
Water Oxygen Map
-1.2 kcal/mol
Surface Type
Molecule
Protein Surface
Protein Cartoon
Exclusion Map
1fjs.protein
Visualize FragMap
Load Ligands
Load MC-SILCS
Exit FragMap tools

CHAPTER

SEVEN

LIGAND POSING AND SCORING ON GFE FRAGMAPS

The power of SILCS lies in the ability to use FragMaps to rapidly evaluate diverse ligands. SILCS-MC is Monte-Carlo (MC) sampling of ligands in translational, rotational, and torsional space in the field of FragMaps. The score of a ligand pose is evaluated as a combination of CGenFF intramolecular energies and the LGFE (Ligand Grid Free Energy), which is the sum of atomic GFEs. While ligand does not "see" the protein, the exclusion map prevents it from sampling areas where no probe or water molecules visited during SILCS simulations. This allows for rapid docking of the ligand while accounting for protein flexibility in a mean- field like fashion as that information is embedded in the FragMaps and the exclusion map. For more details see *[7]*.

Two SILCS-MC presets are available, pose generation and pose refinement, and are described below. Instructions for developing a custom SILCS-MC protocol are provided in a a subsequent section.

7.1 SILCS-MC presets

7.1.1 Pose generation

Pose generation 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 is predefined as a 10 Å sphere and the center of the pocket is taken as either the center of the supplied ligand molecule coordinates or explicitly given by the user. This protocol entails five independent MC runs with the ligand.

This protocol is recommended for ligands with diverse chemotype and unknown binding poses. When the pose of a parent ligand is known and MC- SILCS evaluations are to be performed over a congeneric series, the pose refinement protocol is recommended instead (see below).

To set up and run SILCS-MC pose generation, 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.

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_exhaustive \
    prot=<prot pdb> \
    ligdir=<directory containing ligand mol2/sdf> \
    mapsdir=<directory containing SILCS FragMaps> \
    center="x,y,z"
```

Pose generation spawns five independent single-core serial jobs per ligand and typically takes 30-60 minutes for per ligand. Each run involves a maximum of 250 cycles and a minimum of 50 cycles of Monte Carlo/Simulated Annealing (MC/SA) sampling of the ligand within the defined 10 Å sphere. 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 beginning of each cycle, the ligand will be reoriented within the predefined sphere. When no sphere center is defined by the user, ligand orientation defined in the Mol2/sd file will be used instead as a starting pose at the start of each cycle. 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 PDB file: `3_silcsmc/<run>/pdb/<lig>_mc.<run>.pdb`.

In each run, 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 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. After all five runs are finished, the lowest LGFE pose among all the five runs is output in `3_silcsmc/minconfpdb/<lig>_silcsmc.minconf.pdb`.

7.1.2 Pose refinement

The pose refinement protocol is designed to limit conformational sampling near the ligand input pose supplied by the user. Pose refinement is approprpriate when there is high confidence in the input parent ligand pose and SILCS-MC evaluations are to be performed over a congeneric series. The sphere center for the pocket definition is the center-of-mass from the input ligand pose.

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.

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_local prot=<prot pdb> \
    ligdir=<directory containing ligand mol2/sdf> \
    mapsdir=<directory containing SILCS FragMaps>
```

Pose refinement spawns five independent single-core serial jobs per ligand, nd each run involves a maximum of 10 cycles of MC/SA sampling of the ligand within a 1 Å sphere. The center of the sphere is defined as the center-of-mass of the input ligandpose. Each of cycles 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 1 Å, ii) molecular rotation with a maximum step size of 180 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 PDB file `3_silcsmc/<run>/pdb/<lig>_mc.<run>.pdb`. After all five runs are finished, the lowest LGFE pose among all the five runs is output in `3_silcsmc/minconfpdb/<lig>_silcsmc.minconf.pdb`.

7.1.3 User-defined protocols

Apart from the two presets, users can define their own protocols to drive SILCS- MC sampling of ligand(s) in the field of FragMaps. To do so, copy the `${SILCSBIODIR}/templates/silcs-mc/params_custom.tmpl` to the location where you plan to setup the SILCS-MC calculation and edit this file. The parameters in the bracket, e.g., `<parameter>`, will be replaced later when the program is executed.

The parameters in the angle brackets, e.g., `<parameter>`, will be replaced by the program to a default value when the program is executed. User will be responsible for providing and making sure to using:

```
${SILCSBIODIR}/programs/silcs_mc -h
```

After the desired parameters are set in the `params_custom.tmpl`, use the following command to setup and run SILCS-MC procedure. Each ligand can be stored as a separate `.mol2` or `.sdf` file. Alternatively, all the ligands can be combined into a single `.sdf` file.

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_custom prot=<prot pdb> \
    ligdir=<directory containing ligand mol2/sdf> \
    mapsdir=<directory containing SILCS FragMaps> \
    paramsfile=<params_custom.tmpl>
```

Additionally, the number of runs that will be spawned can also be modified, by using another command-line parameter, `totruns`.

```
${SILCSBIODIR}/silcs-mc/1_run_silcsmc_custom prot=<prot pdb> \
    ligdir=<directory containing ligand mol2/sdf>
    mapsdir=<directory containing SILCS FragMaps>
    paramsfile=<params_custom.tmpl>
    totruns=<# of runs>
```

See below for the detailed description of parameters that need to be defined in the `params_custom.tmpl` file.

- `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.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/par_all36_cgenff.prm`

- `SILCS_RULES <silcs rules file>` (required)

  This rule file is 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_classification_rules_feb16_generic.dat`

  Another variant of this rule file is available at `${SILCSBIODIR}/data/silcs_classification_rules_feb16_specific.dat`.

  When `silcs_classification_rules_feb16_generic.dat` is used, ligand atoms are assigned using six generic classifications in mapping them back to the FragMaps: `APOLAR`, `HBDON`, `HBACC`, `MEOO`, `ACEO`, and `MAMN`. All the atoms in the ligand fall into one of these six categories.

  When `silcs_classification_rules_feb16_specific.dat` is used, ligand atoms are assigned using 13 more specific classifications in mapping them back to FragMaps: `BENC`, `PRPC`, `AALO`, `MEOO`, `FORN`, `FORO`, `IMIN`, `IMIH`, `ACEO`, `MAMN`, `APOLAR`, `HBDON` and `HBACC`. For instance, aromatic ring carbons and aliphatic linker carbons are distinguished as `BENC` and `PRPC`, respectively here, while they are grouped as `APOLAR` atoms in the generic rule file.

- `GFE_CAP <default: 3.0>`

  Maximum unfavorable GFE (kcal/mol) accounted 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.

- `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>`.

  Max 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.

- `SIM_ANNEAL_MOVE_RANGE <default:0.2 9.0 9.0>`

  Max 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.

- 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: 1.0 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 conformation

  When set to FALSE, then the center-of-mass of the ligand is used as the center for the MC simulation. This is useful when the ligand pose in the pocket is well-known.

- ATOM_TO_RESTRAIN: <atom number in sd/mol2>

  When set, a spherical potential is applied to restrained 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. Also, random pocket pose and placement using RANDOM_INIT_ORIENT true is incompatible.

  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`.

- `CLUSTER_RADIUS <default: 0.6 A>`

  Used clustering for ligand binding poses.

- `OUTCLUSTPDBFILE: <output PDB file>`

  This file stores representative cluster conformations.

- `LIGAND_SDF: <ligand sdf>`

  When ligands are in an `.sd` file, use this option. Only if no sdf is supplied from command-line, this line will be read.

- `LIGAND_MOL2: <ligand mol2>`

  When ligands are in mol2 file, use this option. Only if no mol2 is supplied from command-line, this line will be read

  One of these parameters (LIGAND_MOL or LIGAND_SDF) need to be set correctly for SILCS-MC simulation to proceed. Interchanging `LIGAND_MOL2` and `LIGAND_SDF` for the wrong file-types lead to SILCS-MC simulation to not proceed.

- `SILCSMAP <MapType> <map name>` (required)

  Multiple `SILCSMAP` flags can be defined, with each flag pointing to one file. Standard SILCS FragMaps of the following `<MapType>` should be included with the below keywords:

  - EXCL - exclusion map
  - NCLA - zero map for non-classified ligand atoms
  - BENC - benzene carbon
  - PRPC - propane carbon
  - MEOO - methanol alcohol oxygen
  - FORN - formamide nitrogen
  - FORO - formamide oxygen
  - MAMN - methyl ammonium nitrogen
  - ACEO - acetate oxygens
  - AALO - acetaldehyde oxygen
  - IMINH - imidazole donor nitrogen
  - IMIN - imidazole acceptor nitrogen
  - APOLAR - generic non polar : green

  - HBDON - generic donor: blue
  - HBACC - generic acceptor: red

- `OUTMCPDBFILE <output PDB file>` (required)

  This file stores the lowest energy conformation from each cycle of the MC/SA simulation.

- `OUTMCLOGFILE <output log file>`

  This file stores the energy statistics of the lowest energy conformation from each cycle of the MC/SA simulation.

- `OUTMCPDBFILEPREFIX <output PDB file(s)>`

  When reading SD files (with one or more molecules), this file(s) store the lowest energy conformation from each cycle of the MC/SA simulation. Filenames are appended with the name of the ligand(s)

- `OUTMCLOGFILEPREFIX <output log file(s)>`

  When reading SD files (with one or more molecules), this file(s) stores the energy statistics of the lowest energy conformation from each run of the MC/SA simulation. Filenames are appended with the name of the ligand(s)

  If `LIGAND_SDF` is defined, then `OUTMCPDBFILEPREFIX` and `OUTMCLOGFILEPREFIX` need to be defined and need to point to the right files for SILCS-MC to proceed.

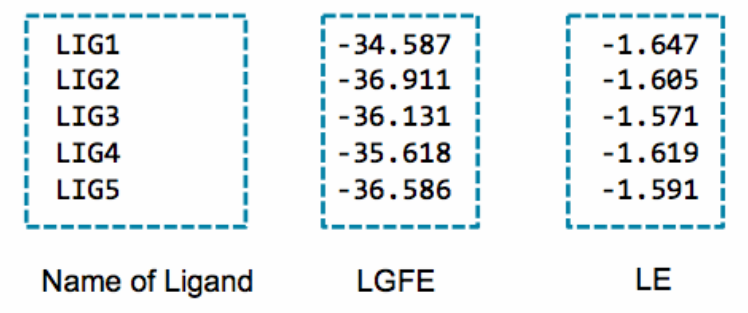
Note: Ligand related parameters such as `LIGAND_MOL2`/`LIGAND_SDF`, `OUTMCPDBFILE`/`OUTMCLOGFILE`, need not be manually set. The script `1_run_silcsmc_custom` will set these parameters appropriately for each of the ligand defined in the `ligdir` directory.

7.1.4 LGFE and best pose retrieval

Once the SILCS-MC simulation is finished, retrieve the LGFE scores for each of the ligands that have been subjected to SILCS-MC using:

```
${SILCSBIODIR}/silcs-mc/2_calc_lgfe_min_avg_sd ligdir=<directory␣
↪containing lig mol2>
```

An example of the output of this script is:



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.

$$LE = \frac{LGFE}{N_{\text{HeavyAtoms}}}$$

7.2 Binding pocket identification

Please contact info@silcsbio.com for more information on this capability.

7.3 Excipient design for biologics

Please contact info@silcsbio.com for more information on this capability.

CHAPTER

EIGHT

SINGLE STEP FREE ENERGY PERTURBATION (SSFEP)

8.1 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 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 large number of changes to a parent ligand, while maintaining useful accuracy for small functional group modifications *[3]*.

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. Zwanzig's FEP formula is used,

$$\Delta G^{\mathrm{env}}_{\mathrm{L1}\to\mathrm{L2}} = -k_{\mathrm{B}}T \ln \left\langle e^{-\beta \Delta E} \right\rangle_{\mathrm{L1}} \tag{8.1}$$

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:

$$\Delta E = E_{L2-\mathrm{env}} - E_{L1-\mathrm{env}}$$

The environment `env` in each system is defined as all non-ligand atoms. As the environment is constant between the two ligands, the internal environmental energy cancels exactly during the computation of ΔE. In addition, as the difference between L1 and L2 involves a very small number of heavy atom modifications, we expect any differential intra-ligand energy terms to also cancel exactly between the solution and protein environments. Therefore, once $\Delta G^{\mathrm{protein}}_{L1\to L2}$ and $\Delta G^{\mathrm{water}}_{L1\to L2}$ are computed according to Eq. (8.1), the relative binding free energy is given by

$$\Delta\Delta G^{\mathrm{bind}}_{L1\to L2} = G^{\mathrm{protein}}_{L1\to L2} - G^{\mathrm{water}}_{L1\to L2}$$

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.

In a recent study *[4]*, 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. SSFEP was able to produce comparable results to ful FEP while requiring a small fraction of the computational resources.

8.2 Usage

The following usage details are provided for completeness. **We strongly recommend using the SilcsBio GUI to set up, run, and analyze SSFEP calculations** as described in *Quickstart Guide: Graphical User Interface*.

8.2.1 SSFEP simulation setup

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.

8.2.2 Chemical group transformations

Create a text file `modifications.txt` with instructions listing the desired modifications to the 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 8.1: Available modification types.

An example ligand modification input file is provided in `examples/ssfep/modification.txt` file. For help on how to define these operations, run the following command:

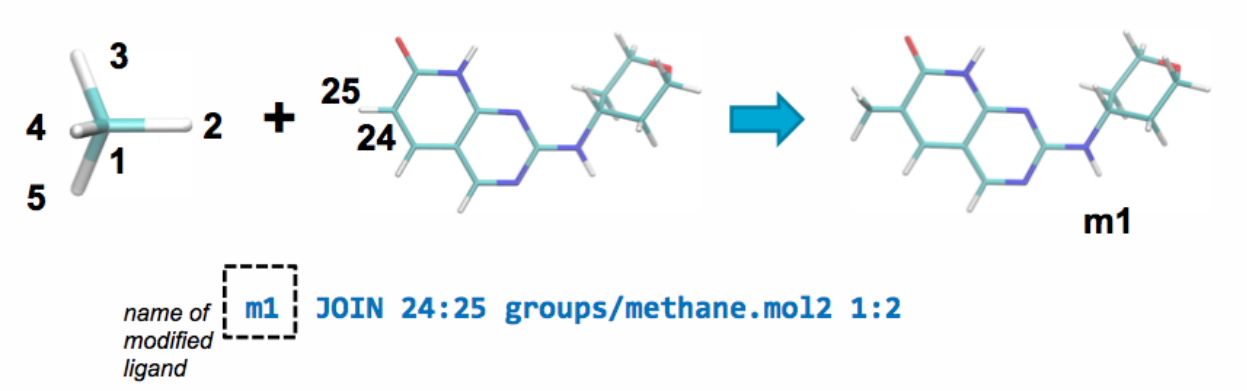
```
${SILCSBIODIR}/programs/chemmod -h
```

Access the groups directory from the following location to look at all the possible fragments that can be added/joined to the parent ligand:

```
${SILCSBIODIR}/data/groups
```

Modifications JOIN example

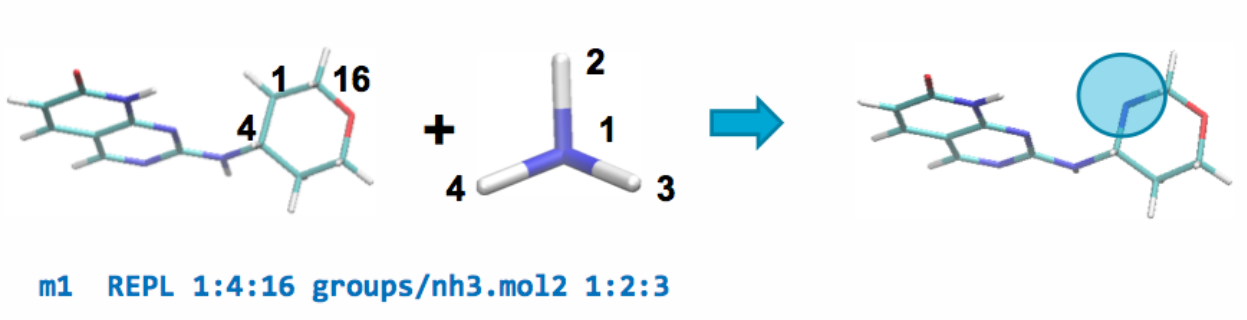
A **JOIN** operation can be performed between the parent and a `ch4.mol2` fragment by adding the following line to `modifications.txt`:



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.

Modifications REPL example

A **REPL** operation can be performed between the parent and a `ch4.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.

Modifications 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 (atom index 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 ssfep_lig_decoration_master_modification.txt
```

Copy `ssfep_lig_decoration_master_modification.txt` from the `${SILCSBIODIR}/data/` folder. This master modification list contains 70 commonly-used modifications. Be careful to pay attention to chemistry; if some of these modifications are not suitable for a site, you can comment them out using `!` at the beginning of that line.

This single entry in your `modifications.txt` will generate 70 separate modifications to the parent ligand, each with a prefix `m1_`

8.3 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.

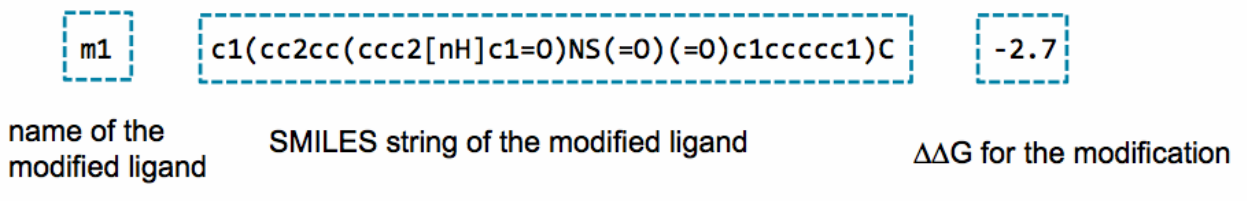
```
${SILCSBIODIR}/ssfep/3a_setup_modifications prot=<Protein PDB> lig=
↪<Ligand Mol2/SDF File> mod=modifications.txt
```

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

NINE

CHARMM GENERAL FORCE FIELD (CGENFF)

9.1 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 *[1]*. The parametrization philosophy behind the force field focuses on quality at the expense of transferability, with the implementation concentrating on an extensible force field.

CGenFF uses the CHARMM additive potential energy function to calculate the energy as a function of the Cartesian coordinates of the system, as shown below.

$$\begin{aligned}&\mathrm{Intramolecular(internal, bonded terms)}\\&\sum_{\text{bonds}} K_b(b-b_0)^2 + \sum_{\text{angles}} K_\theta(\theta-\theta_0)^2 + \sum_{\text{dihedrals}} K_\phi(1+\cos(n\phi-\delta))\\&+ \sum_{\text{improper}} K_\psi(\psi-\psi_0)^2 + \sum_{\text{Urey-Bradley}} K_{\mathrm{UB}}(r_{1,3}-r_{1,3;0})^2\end{aligned}$$

$$\begin{aligned}&\mathrm{Intermolecular(external, nonbonded terms)}\\&\sum_{\text{nonbonded}} \frac{q_i q_j}{4\pi D r_{ij}} + \epsilon_{ij}\left[\left(\frac{R_{\min,ij}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{\min,ij}}{r_{ij}}\right)^{6}\right]\end{aligned}$$

The bonded or intramolecular part 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 or intermolecular 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, $R_{\min,ij}$ is the radius, and r_{ij} is the distance between i and j.

It is apparent that a simulation on any system of practical 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 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.

CGenFF comes with 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. 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 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 *[2]*.

9.2 Usage

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}/programs/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 outpu. It is up to the user to 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 1.0.0
* For use with CGenFF version 3.0.1
*

read rtf card append
... topology section ...

END

read param card flex append
... parameter section ...
```

(continues on next page)

(continued from previous page)

```
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
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 initial atomic 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 charges, each entry in parameter 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.

9.3 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.

9.4 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 >200) 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 of the molecule of interest.

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 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. 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.

9.4.1 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 restrainted 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 rapidily.

8. Setting the RMS energy difference to not redo the least-squares fitting using different multiplicities.

```
rmse_cutoff=<RMS energy cutoff to determine if addiional␣
↪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 perfomed 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.

9.4.2 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
```

(continues on next page)

(continued from previous page)

```
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 optimizating 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=200
```

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 ...
```

(continues on next page)

(continued from previous page)

```
##############################################
##############################################

Nothing to optimize based on the penalty score threshold of  50

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
```

(continues on next page)

(continued from previous page)

```
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
ATOM      14 N    NONA     1       0.613  -0.617   1.406  1.00  0.00
 ↪ NG2R50
```

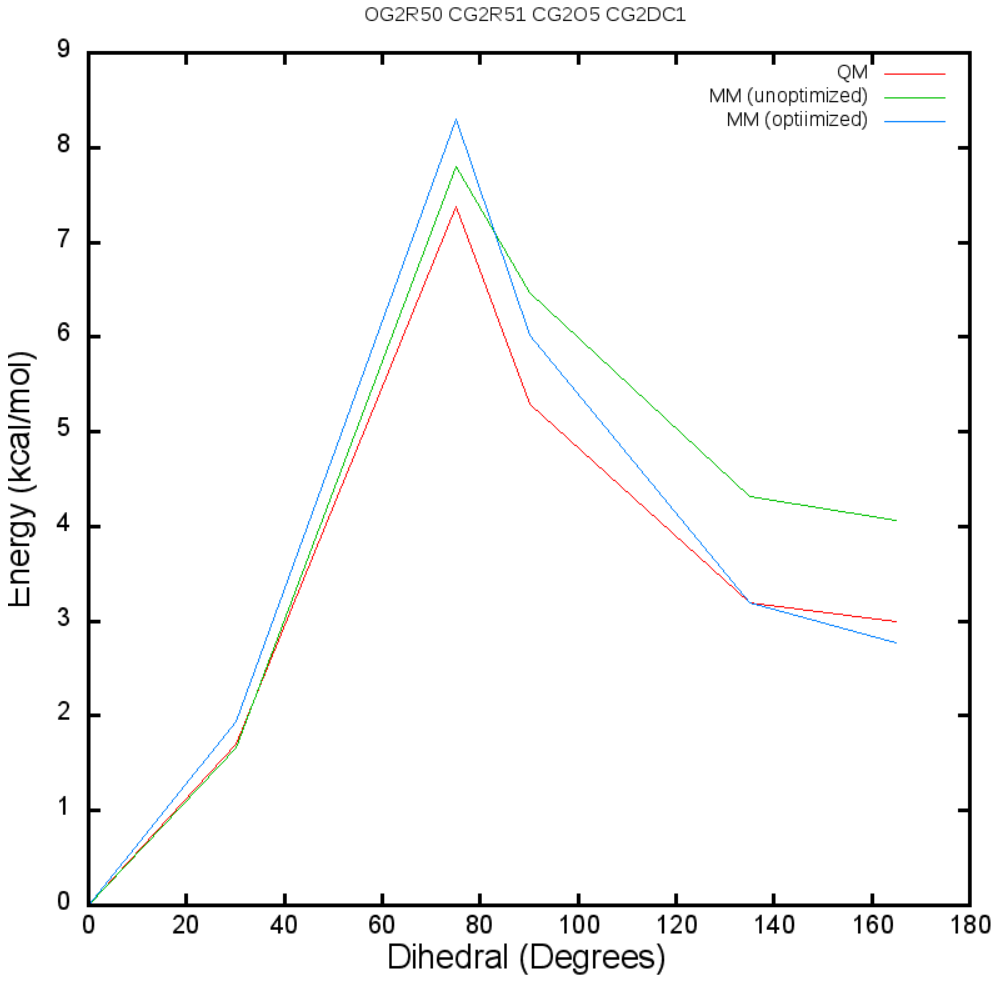
Following the identification of the dihedrals to the fitted, the Optimizer performs a complete molecular mechanics (MM) minimization of the molecule using the default CGenFF force field parameters. Next, each identified dihedral is independently rotated through 360 degrees with the specified step-size. 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 parallely 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
```

9.4.3 Performing dihedrals 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 &
```

9.4.4 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 (eg. 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 admistrator 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.

9.4.5 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.

9.4.6 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 (eg. 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 (ie. 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 finshed, the new parameters may be combined and added to your the CGenFF parameter files.

During parameter fitting, the program intially 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 (Under development) ${SILCSBIODIR}/cgenff-optimizer/refit_par_with_subsurface dih_qm_<dihno>.dat dih_mm_<dihno>.dat"

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" (eg. psi4_inp_xxxx_1_23.inp) maybe submitted directly to complete those jobs followed by resubmission of the cgenff-optimzer command from the parent directory.

BIBLIOGRAPHY

[1] 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.

[2] 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.

[3] 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.

[4] 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.*, October 2016.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.