G-LoSA Search Steps in Sherlock

This page contains the instructions for running G-LoSA search against a database. The general pipeline to run G-LoSA is broken down into preparing a library (optional if using downloaded library), running the search, and processing the results.

Starting Sherlock

```
sdev # starts interactive job session (optional)
ml load java/[version] # loads java
```

Library and Protein Prep

1. Convert target protein from .mae to .pdb with PyMol and extract protein binding pocket
    a. Load the target protein AND associated ligand .mae file into pymol
    b. Click on the ligand in the viewer to select the molecule
    c. Use the console to run

        `select pocket, br. all near_to 8 of sele`

        This command selects all residues within 8 angstroms from the ligand. To use another distance, replace 8 with the desired distance.
    d. Save the selection ‘pocket’ as the target protein binding pocket to be used in G-LoSA search.
2. Strip PDB - Check the pdb file of the target protein binding pocket. If the file contains any lines that do not start with ATOM or HETATM, then that line should be deleted, or else G-LoSA will not be able to parse the file. You can manually delete the lines or use the java program StripPDB.java in Scripts/

3. Only follow steps 4 and 5 if you want to use your own library. If using the library provided by G-LoSA, these steps are unnecessary.
4. Build the Binding Site Library if no library exists: This program takes a list of pdb codes (of binding site templates), and the associated pdb files, and outputs a directory of the individual chains of all binding site templates. It is important to have template ligands in the folder containing the binding site template pdbs (they don't have to be in the list because they are not being used except to be included in the output folder). Note: Refer to the Template Library Format section if this is confusing.
    a. Input:
        i. Text list of binding site template pdb files, line separated
        ii. Directory of where binding site template and template ligand pdbs are located
        iii. Directory of where the library should be made (output directory) (clarify?)
    b. Output:
        i. Filled Directory containing Binding Site Library
    c. Example:

```
java BuildLigandBSStructureLibrary template_list.txt /path/to/templates/ /desired/lib/path
```

5. Generate Chemical Feature (CF) Files for Binding Sites: This program generates Chemical Feature Files for a list of binding site templates. CF files are necessary to run G-LoSA, so every template must have a generated CF file.
    a. Input:
        i. Text list of binding site template pdb files, line separated
        ii. Directory of where binding site template pdbs are located
        iii. Directory of where the CF files should be made (output)
    b. Output:
        i. Filled Directory containing Chemical Feature Files for each Template in library
    c. Example:

```
java GenerateCFFiles template_list.txt /path/to/templates/  /output/folder/
```

Formatting the Template Library

The binding site template library has 4 main types of files:

1. Binding site template pdb
2. Binding site template ligand pdb
3. Binding site template CF file
4. Text list(s) of binding site template pdb codes
    a. Note: This is used if you want to partition the library into multiple sections in order to speed things up (running 10 jobs of 1,000 templates vs 1 job of 10,000 templates). G-LoSA uses the text lists to iterate through the search so by splitting up the lists you can run parallel jobs manually.

There are multiple ways to format the binding site template library to run G-LoSA search. I formatted the library in the following way:

▼ G-LoSA Folder

- ▼ Text list(s) of binding site teamplte pdb codes (file type 4)
- ▼ Folder of template and template ligand pdbs (file type 1/2)
- ▼ Folder of template CF files (file type 3)
- ▼ One G-LoSA Search Job Folder (refer to job folder structure to see how a job is structured)

It should look something like:

G-LoSA Job Folder Structure

G-LoSA Job Folders are folders that contain the input and results to run G-LoSA jobs. I structured them in the following way:

▼ Parent Folder (Name of target protein)

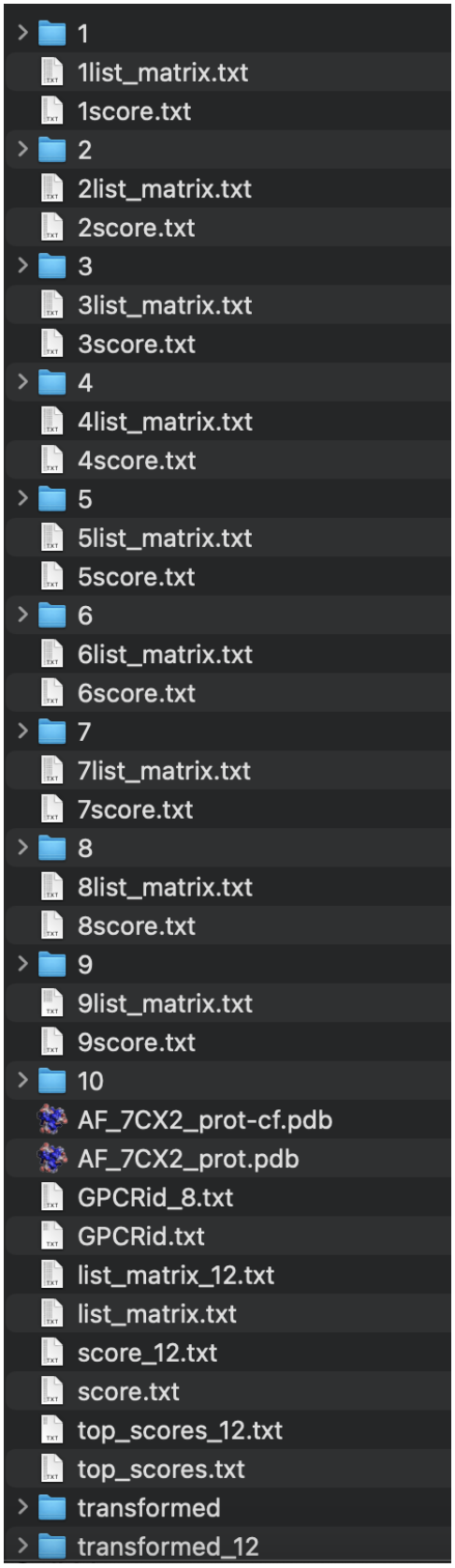
- ▼ Folders named 1 through 10 (contains batch job information for each section of the binding site library - the library is split into 10 sections)
    - ▼ [num].inp - input file for that section of the library
    - ▼ [num].sbatch - sbatch file for that section of the library
    - ▼ Any other file are those produced by slurm output for temp files for running GLoSA
- ▼ [transformed] (directory to hold transformed alignments of the top binding sites)

- ▼ [target_protein].pdb - target protein file
- ▼ [target_protein]-cf.pdb - Chemical features file of target protein
- ▼ [num]score.txt - score.txt output file for [num] section of the library
- ▼ [num]list_matrix.txt - matrix transformation info for binding sites that meet the cutoff score in [num] section of the library
- ▼ score.txt - list of the scores for the entire library, made by merging all the [num]score.txt files
- ▼ list_matrix.txt - alignment info for the entire library, made by merging all the [num]score.txt files
- ▼ top_scores.txt - Top binding sites listed and their scores

Example (AF pdb code 7CX2):

- ▼ AF_7CX2
  - ▼ 1
    - ▼ 1.inp
    - ▼ 1.sbatch
  - ▼ 2
    - ▼ 2.inp
    - ▼ 2.sbatch
  - ▼ 3
    - ▼ 3.inp
    - ▼ 3.sbatch
  - ▼ 4
    - ▼ 4.inp
    - ▼ 4.sbatch
  - ▼ 5
    - ▼ 5.inp
    - ▼ 5.sbatch

- 6
  - 6.inp
  - 6.sbatch
- 7
  - 7.inp
  - 7.sbatch
- 8
  - 8.inp
  - 8.sbatch
- 9
  - 9.inp
  - 9.sbatch
- transformed
  - [transformed bs/lig pairs]
- AF_7CX2_prot.pdb
- AF_7CX2_prot-cf.pdb
- [1-num]score.txt - score.txt output file for [num] section of the library
- [1-num]list_matrix.txt - matrix transformation info for binding sites that meet the cutoff score in [num] section of the library
- score.txt - list of the scores for the entire library, made by merging all the [num]score.txt files
- list_matrix.txt - alignment info for the entire library, made by merging all the [num]score.txt files
- top_scores.txt - Top binding sites listed and their scores
- GPCRid.txt - Top binding sites labelled by whether they are from GPCRs or not


1
1list_matrix.txt
1score.txt
2
2list_matrix.txt
2score.txt
3
3list_matrix.txt
3score.txt
4
4list_matrix.txt
4score.txt
5
5list_matrix.txt
5score.txt
6
6list_matrix.txt
6score.txt
7
7list_matrix.txt
7score.txt
8
8list_matrix.txt
8score.txt
9
9list_matrix.txt
9score.txt
10
AF_7CX2_prot-cf.pdb
AF_7CX2_prot.pdb
GPCRid_8.txt
GPCRid.txt
list_matrix_12.txt
list_matrix.txt
score_12.txt
score.txt
top_scores_12.txt
top_scores.txt
transformed
transformed_12

Running GLoSA Search

1. Run G-LoSA Search: This is the main program that runs G-LoSA with the target AF binding pocket and the library of templates. The only input is a file with ending .inp - the format of that file is displayed below
    a. Input
        i. inp file in format:
            1. pdb of target protein
            2. chemical feature file of target protein
            3. surface residue pdb (optional)
            4. surface residue chemical feature file (optional)
            5. directory of template library
            6. list of templates
            7. directory of template chemical features
            8. location of glosa
            9. cutoff GA-score
    b. Output
        i. score.txt - GA-score of all binding sites
        ii. matrix.txt - transformation matrix information
        iii. list_matrix.txt - transformations for binding sites that meet the cutoff GA-score
    c. Example:

    Input file:

```
query protein structure file (PDB)
> /home/user/project/[ref].pdb
chemical feature (CF) file for query protein structure
> /home/user/project/[ref]-cf.pdb
surface residues file for query protein structure (PDB) (n: dont use this option)
> /home/user/project/[ref]-surface.pdb
chemical feature (CF) file for surface residues (n: dont use this option)
> /home/user/project/[ref]-surface-cf.pdb
directory of PDB local structure library
> /home/user/project/structure_library/
list of local structures in structure library
```

```
> /home/user/project/structure_library/template_list.txt
directory where CF files for library local structures are saved
> /home/user/project/structure_library_cf/
G-LoSA execution file
> /home/user/project/glosa
cutoff GA-score for writing rotation matrix
> 0.6
```

Command:

```
java GLoSASearch [input].inp
```

Running Sherlock Job - batch file and command

```
#!/bin/bash
#SBATCH --job-name=[job_name]
#SBATCH --time=5:00:00
#SBATCH -p owners
#SBATCH -c 2
#SBATCH --mem=16GB

ml load java/11.0.11

java GLoSASearch [input].inp
```

```
sbatch [job].sbatch
```

2. Get Top Binding Site Templates: If the template library is large, filtering the top templates out is useful. This program takes in the result file of the G-LoSA search as well as threshold values for GA-score (G-LoSA's scoring function) and the placement cutoff of the template (Top 10, 20, etc.). It then returns a list of the top templates based on the input cutoffs.
    a. Input
        i. result file (score.txt)
        ii. number - threshold of top N templates
        iii. Threshold GA-score (filter by this or top N)
        iv. Output file

b. Output

   i. file of the filtered templates

c. Example:

```
java GetTopTemplates score.txt 10 0.6 top10.txt
```

3. Get Alignments between Binding Site Templates and target protein: Using the top templates list generated by GetTopTemplates, transformations of the templates that will best align them with the target AF binding site can be generated using GetAlignedTemplates.

   a. Input

      i. list of templates (or more often top templates generated by GetTopTemplates)

      ii. directory of the template library

      iii. rotation matrix file from results (list_matrix.txt)

   b. Output

      i. directory containing structures of aligned template pdbs

   c. Example:

```
java GetAlignedTemplates template_list.txt lib/ list_matrix.txt transformed_structures/
```

4. View in PyMol by loading in the reference protein and the transformed structures. If the data is on Sherlock and PyMol is being run locally, it may be convenient to mount the G-LoSA job folder using sshfs and then loading the protein models:

   `load /path/to/target/model.pdb` to load the target model

   `loadall /path/to/transformed/*` to load the transformed top templates

Handy Notes

- If partitioning the library into multiple sections and running multiple jobs, there will be multiple `score.txt` and `list_matrix.txt` files. To continue analysis, it is easiest to merge these. A simple way to do so is to use `cat` (i.e. `cat *score.txt > score.txt`)

- Creating a Job folder from scratch may be time consuming (especially if partitioning the library into multiple sections). To keep the same G-LoSA search settings and run the search on a new protein, one can copy and paste an old job folder, delete the results files for that job, and modify the inputs to run with the new protein, using these commands:
    - `find . -maxdepth 2 -type f -name "slurm*" -delete` to delete the slurm output files from the previous job
    - `find . -maxdepth 2 -type f -exec sed -i 's/AF_old_pdb/AF_new_pdb/g' {} \;` to replace all instances of `[old_pdb]` to `[new_pdb]` . Replace the old pdb code that was analogous to the AF model of the previous job to the new one.

Debugging Log

- Some pdb files need filtering to be read by GLoSA
    - created script to filter pdb to GLoSA format (StripPDB)
- cf files should be in a separate folder than template binding structures (forgot why)
- ligands should not be on the search list (to save time)
- score of 0 indicates error occurred running GLoSA search
    - Most likely seg fault from running GLoSA
- TODO: modify GLoSA search output so it doesn't overwrite the same file every time you run