Protein Folding

Bold text means that these files and/or this information is provided.

Italicized text means that this material will NOT be conducted during the workshop.

`Fixed width text means you should type the command into your terminal.`

If you want to try making files that already exist (e.g., input files), write them to a different directory!

Tutorial 1: Protein Folding

This tutorial presents a protein folding benchmark experiment. Bacteriophage T4 lysozyme is a soluble protein that hydrolyzes peptidoglycan and releases the virus from its bacterial host. The protein was crystallized at a resolution of 1.7 angstroms and the resulting structure was submitted to the Protein Data Bank (PDB) under accession number 2LZM. In this tutorial, you will reconstruct the structure of bacteriophage T4 lysozyme using *ab initio* protein folding. At the end of the tutorial, the results from this benchmark experiment will be compared to the native structure available from the PDB.

1. Prepare your working directory. You will work in this directory for the rest of the tutorial.

   Create a directory in the protein_folding directory called my_files and switch to that directory.

   ```
   mkdir my_files
   cd my_files
   ```

2. Prepare your input files.

   1. Save your protein sequence to a file using FASTA format.
      The 2LZMA.fasta file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

      1. Get the sequence in FASTA format from NCBI.
         1. Go to http://www.ncbi.nlm.nih.gov/protein/.
         2. Type in 2LZMA in the search bar at top.
         3. Click on the "FASTA" link.
         4. Open a text editor, such as gedit. Copy all the sequence information, including the line beginning with ">", into a new file.
         5. Remove the first 57 residues of the sequence so that it begins with the sequence ITKDE.
         6. Save the file as 2LZMA.fasta to the my_files directory.

   2. Prepare a PDB of native structure.
      The clean and renumbered 2LZMA.pdb file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

      1. Get the native PDB structure.
         1. Go to http://www.pdb.org/.
         2. Search for 2LZM.
         3. Click "Download Files" and "PDB File (Text)" in the top right hand corner by the PDB ID.
         4. Save the PDB file as 2LZM.pdb.
         5. The file may end up downloading to your Downloads directory. If so, move it to your working directory using this command:

            ```
            mv ~/Downloads/2LZM.pdb .
            ```

      2. Clean 2LZM.pdb so that only ATOM records remain:

         ```
         python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/clean_pdb.py 2LZM A
         ```

- NOTE: This outputs 2 files, 2LZM_A.fasta and 2LZM_A.pdb.

3. In a text editor, remove the first 57 residues from 2LZM_A.pdb that it begins with the sequence ITKDE.
4. Renumber 2LZM_A.pdb:

```
python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/pdb_renumber.py \
2LZM_A.pdb 2LZMA.pdb
```

3. Prepare 3mer and 9mer fragment libraries.

The 2LZM fragment libraries (aa2LZMA03_05.200_v1_3 and aa2LZMA09_05.200_v1_3), secondary structure prediction (2LZMA.psipred_ss2 and 2LZMA.jufo_ss2), and checkpoint (2LZMA.checkpoint) files are provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

1. Make fragment libraries using Robetta (for the purposes of this workshop).
   1. If you are an academic or non-profit user of Rosetta, make sure you're registered at http://robetta.bakerlab.org/.
   2. Under "Services", "Fragment Libraries" click "Submit".
   3. Type 2LZMA under "Target Name".
   4. Copy/paste all the text in 2LZMA.fasta into the "Paste Fasta" field.
   5. If you are benchmarking, you will want to check "Exclude Homologues".
   6. Click "Submit".
   7. You can see your position in the queue by clicking "Queue" under "Fragment Libraries". This should not take very much time unless the queue is long.
   8. When your job is finished, you should receive an email with a link called "Result Details". Click on this link. Then, click on "DOWNLOADS".
   9. Your fragment files should be called aa2LZM_03_05.200_v1_3 and aa2LZM_09_05.200_v1_3. However, it is wise to keep all of the files, especially the checkpoint, psidpred_ss2, and jufo_ss2 files. Right click on the files, and save the files to your working directory.
2. *Or, make fragment libraries using make_fragments.pl (not covered by this tutorial; see Appendix).*

4. Prepare the topology broker setup file.

The topology_broker.tpb setup file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

1. Save the following text as topology_broker.tpb in your working directory:

```
CLAIMER SequenceClaimer
FILE ./2LZMA.fasta
END_CLAIMER
```

5. Prepare the topology broker options file.

The 2LZM_broker.options file is already provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

1. Rosetta ignores lines beginning with # (these are treated as comments).
2. Avoid mixing tabs and spaces. Be consistent in your formatting (tab-delimited or colon-separated).
3. Save the file in your working directory.

6. Prepare the core residue specification file.

The 2LZMA_core.txt file is already provided for you in the ~/rosetta_workshop/tutorials/protein_fold directory.

1. The 2LZMA_core.txt file tells Rosetta over which residues to compute CA-RMSD, easing the analysis burden. Copy it from the input_files directory to the current directory.

```
cp ../input_files/2LZMA_core.txt .
```

3. Run the Rosetta topology broker using the Minirosetta application.

1. Make sure all the filenames and paths in the options file are correct.

2. Make sure you are in your working directory.
3. Type the following command line:

```
~/rosetta_workshop/rosetta/main/source/bin/minirosetta.default.linuxgccrelease \
@2LZM_broker.options
```

   - NOTE: This will take 10-20 minutes per structure.

4. Analyze your data.

   Example data is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/output_files/example_data/ directory.

   1. For practice, we will be analyzing data that has already been generated.
      1. Create a directory for analysis of your data and switch into that directory.

         ```
         mkdir data_analysis
         cd data_analysis
         ```

      2. Copy the files from the output_files/example_data/ directory.

         ```
         cp ~/rosetta_workshop/tutorials/protein_folding/output_files/example_data/* .
         ```

   2. Identify the best scoring models.
      1. Format the score and RMS data for graphing using the score_scatter_plot.py script.

         ```
         python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/score_scatter_plot.py \
         --x_axis rms_core --y_axis score --silent 2LZM_broker_01.out 2LZM_models.table
         ```

         - **The 2LZM_models.table file output is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/output_files/example_analysis/ directory.**

      2. Sort the 2LZM_models.table by the score column from lowest to highest.

         ```
         sort -nk3 2LZM_models.table > 2LZM_models_sorted.table
         ```

      3. Identify the top 5-10 models by score.

         ```
         head -n 10 2LZM_models_sorted.table
         ```

   3. Extract the top scoring models from the binary silent file.

      ```
      ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.default.linuxgccrelease \
      -in:file:silent 2LZM_broker_01.out -in:file:fullatom -out:pdb -out:file:fullatom \
      -in:file:tags S_0011 S_0013 S_0008 S_0024 S_0006 S_0005 S_0016 S_0002 S_0018 S_0014
      ```

   4. Look at best-scoring models by opening them in PyMol or the molecular graphics program of your choosing.

      ```
      pymol *.pdb
      ```

   5. Generate a Score vs. RMSD plot.
      1. Determine which model is the lowest-scoring. It is common practice to use the lowest-scoring model as a surrogate for the native structure when analyzing experimental folding results.

         ```
         head -n 1 2LZM_models_sorted.table
         ```

      2. The 2LZMA_core.txt file tells Rosetta over which residues to compute CA-RMSD. Copy it from the input_files directory.

         ```
         cp ~/rosetta_workshop/tutorials/protein_folding/input_files/2LZMA_core.txt .
         ```

      3. Rescore the models, computing RMSD to the lowest-scoring model.

         ```
         ~/rosetta_workshop/rosetta/main/source/bin/score_jd2.default.linuxgccrelease \
         -in:file:silent 2LZM_broker_01.out -in:file:silent_struct_type binary \
         -in:file:fullatom -out:file:silent 2LZM_all_models_rescored_silent.out \
         -out:file:silent_struct_type binary -out:file:fullatom \
         -evaluation:rmsd S_0011_0001.pdb _core_low 2LZMA_core.txt
         ```

      4. Make a table of the scores and RMSDs of your models:

```
grep SCORE 2LZM_all_models_rescored_silent.out | \
awk '{print($NF"\t"$(NF-1)"\t"$2)}' > 2LZM_rescore_vs_rmsd.table
```

5. Make a scatter plot using the program of your choosing. This plot gives you an idea of the relationship between the Rosetta energy of a model and its quality, or accuracy. There are many ways to generate this scatter plot, among them:
   1. For this tutorial use the provided script to run a series of R commands. This script will output the score_vs_rmsCoreLow plot as a PDF file (2LZM_rescore_vs_rmsCoreLow.pdf). The -100 is the maximum energy (worst) which will be ploted. Manually specifying it keeps the plot from being compressed by outliers, but the value would need to be adjusted for different proteins.

      ```
      Rscript \
      ~/rosetta_workshop/tutorials/protein_folding/scripts/score_vs_rmsCoreLow.R \
      2LZM_rescore_vs_rmsd.table 2LZM_rescore_vs_rmsCoreLow.pdf -100 \
      2LZM_score_vs_rmsCoreLow
      ```
   2. *Or, use R (http://www.r-project.org) to make these plots. This workshop does not cover the use of R, but example commands are available in the Appendix.*
   3. *Or, read in the 2LZM_score_vs_rmsd.table file into a graphing spreadsheet software (like Excel) and make an X-Y scatter plot.*
6. Review the resulting scatter plot. If a quality model has been generated you should observe a funnel-shaped distribution with low-scoring models having low RMSD values and incorrect models with higher RMSD values having higher scores.

Tutorial 2: Protein Folding with Restraints

1. Prepare your working directory. You will work in this directory for the rest of the tutorial.
   1. Create a directory in the protein_folding directory called my_files_cst and switch to that directory.

      ```
      cd ~/rosetta_workshop/tutorials/protein_folding/
      mkdir my_files_cst
      cd my_files_cst
      ```
2. Prepare your input files.
   1. Generate the constraints (cst) file.
      The 2LZM_dist_w1.cst file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.
      1. Copy the cst file to your working directory.

         ```
         cp ../input_files/2LZM_dist_w1.cst .
         ```
      2. Review the cst file by opening it in a text editor.
      3. The cst file has the basic format:

         ```
         #cst type atom1 resno1 atom2 resno2 function EPR potential dist weight bin_size
         AtomPair   CB     32    CB     36   SPLINE   EPR_DISTANCE  16.0   1.0   0.5
         AtomPair   CB     59    CB     74   SPLINE   EPR_DISTANCE  19.0   1.0   0.5
         AtomPair   CB     62    CB     71   SPLINE   EPR_DISTANCE  19.0   1.0   0.5
         AtomPair   CB     62    CB     74   SPLINE   EPR_DISTANCE  25.0   1.0   0.5
         AtomPair   CB     63    CB     74   SPLINE   EPR_DISTANCE  14.0   1.0   0.5
         ```
      4. Another common type of constraint score is a bounded quadratic penalty:

         ```
         #cst type atom1 resno1 atom2 resno2 function LowerBound Upper StDev comment
         AtomPair   CB     32    CB     36   BOUNDED    11.5     21.5   1.0  NOE ;dist
         AtomPair   CB     59    CB     74   BOUNDED    11.5     21.5   1.0  NOE ;dist
         AtomPair   CB     62    CB     71   BOUNDED    11.5     21.5   1.0  NOE ;dist
         AtomPair   CB     62    CB     74   BOUNDED    12.5     27.5   1.0  NOE ;dist
         AtomPair   CB     63    CB     74   BOUNDED     1.5     16.5   1.0  NOE ;dist
         ```

2. Generate the options file. This file provides the options needed to fold soluble proteins, and also contains the Rosetta options needed specifically for folding proteins with restraints (EPR distance restraints in this case).

   The 2LZM_broker_cst.options file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.

   1. Try generating this options file on your own. Rosetta ignores lines beginning with # (these are treated as comments).
   2. Avoid mixing tabs and spaces. Be consistent in your formatting (tab-delimited or colon-separated).
   3. Save the file in your working directory.

3. Copy the remaining files needed for protein folding from the input_files directory.

```
cp ../input_files/2LZMA.pdb .
cp ../input_files/2LZMA.fasta .
cp ../input_files/aa2LZMA03_05.200_v1_3 .
cp ../input_files/aa2LZMA09_05.200_v1_3 .
cp ../input_files/2LZMA.psipred_ss2 .
cp ../input_files/topology_broker_cst.tpb .
cp ../input_files/2LZMA_core.txt .
```

3. Run the topology broker to *de novo* fold the input protein guided by experimental restraints.

   1. Make sure all the filenames and paths in the options file are correct.
   2. Make sure you are in your working directory for the constraints run.
   3. Type the following command line:

```
~/rosetta_workshop/rosetta/main/source/bin/minirosetta.default.linuxgccrelease \
@2LZM_broker_cst.options > 2LZM_broker_cst.log &
```

      - NOTE: This will take 15-20 minutes per structure.

4. Analyze your data.

   Example output files are provided for you in the ~/rosetta_workshop/tutorials/protein_folding/output_files_cst/example_data directory.

   1. For practice, we will be analyzing data that has already been generated.
      1. Create a directory for analysis of your data and switch into that directory.

```
mkdir data_analysis_cst
cd data_analysis_cst
```

      2. Copy the files from the example_data directory.

```
cp ~/rosetta_workshop/tutorials/protein_folding/output_files_cst/example_data/* .
```

   2. Identify the best scoring models.
      1. Format the score and RMS data for graphing using the score_scatter_plot.py script.

```
python2.7 ~/rosetta_workshop/rosetta/tools/protein_tools/scripts/score_scatter_plot.py \
--x_axis rms_core --y_axis score --silent 2LZM_broker_cst_01.out 2LZM_cst_models.table
```

         - **The 2LZM_cst_models.table file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/output_files_cst/example_analysis/ directory.**

      2. Sort the 2LZM_cst_models.table by the score column from lowest to highest.

```
sort -nk3 2LZM_cst_models.table > 2LZM_cst_models_sorted.table
```

      3. Identify the top 5-10 models by score.

```
head -n 10 2LZM_cst_models_sorted.table
```

   3. Extract the top scoring models from the binary silent file.

```
~/rosetta_workshop/rosetta/main/source/bin/score_jd2.default.linuxgccrelease \
-in:file:silent 2LZM_broker_cst_01.out -in:file:silent_struct_type binary \
-in:file:fullatom -out:pdb -out:file:fullatom \
-in:file:tags S_0003 S_0004 S_0012 S_0022 S_0025 S_0016 S_0007 \
S_0024 S_0002 S_0020
```

4. Re-score the models in order to take the atom_pair_constraint score into account.
   1. The default talaris2014 weights file doesn't add constraint scores. You have to use a version with the constraints enabled (talaris2014_cst).
      - **The talaris2014_cst.wts file is provided for you in the ~/rosetta_workshop/tutorials/protein_folding/input_files/ directory.**
   2. Score your models using these scoring weights.

```
~/rosetta_workshop/rosetta/main/source/bin/score_jd2.default.linuxgccrelease \
-in:file:silent 2LZM_broker_cst_01.out -in:file:silent_struct_type binary \
-in:file:fullatom -out:file:silent 2LZM_broker_cst_rescore.out \
-out:file:silent_struct_type binary -out:file:fullatom \
-evaluation:rmsd S_0003_0001.pdb _core_low 2LZMA_core.txt \
-constraints:cst_fa_file 2LZM_dist_w1.cst -constraints:cst_fa_weight 4 \
-constraints:epr_distance -score:weights talaris2014_cst -overwrite
```

5. Analyze how well the models satisfy the restraints.
   1. Create a score_vs_rmsCoreLow table:

```
grep SCORE 2LZM_broker_cst_rescore.out | \
awk '{print($2"\t"$14"\t"$(NF-1)"\t"$NF)}' > \
2LZM_broker_cst_score_vs_rmsCoreLow.table
```

   2. You can sort the models by atom_pair_constraint score and see which models satisfy the restraints the best. For example, for 25 restraints weighted by a factor of 4 and scored with the RosettaEPR knowledge-based potential, the best score (100% of restraints satisfied) is -100.00 REU. Often, you want to filter models by some combination of total score and restraint score (see Hirst et al., J. Struct. Biol. 2011).

```
sort -nk2 2LZM_broker_cst_all_score_vs_rmsCoreLow.table | \
head -n10 > top10_atom_pair_constraint_score.txt
```

      - *NOTE: Sometimes, it is useful to plot total score (y-axis) vs. atom_pair_constraint score and look at models that are both low-scoring and have good (low) restraint scores.*
      - *NOTE: Scripts for generating score vs. rmsCore, score vs. rmsCoreLow, and score vs. atom_pair_constraint figures can be found in the ~/rosetta_workshop/tutorials/protein_folding/scripts directory.*
   3. You can also see how much the restraints are violated in terms of distance for each restraint in each PDB file.
      1. In a bash shell, execute:

```
for pdb in `ls *.pdb`; do
    ~/rosetta_workshop/tutorials/protein_folding/scripts/calc_exp_viol_spline.pl \
    ${pdb} 2LZM_dist_w1.cst 25 > ${pdb}_cst_viol.txt
done
```

      2. To get the summary of violations for each PDB, execute:

```
for file in `ls *cst_viol.txt`; do
    tail -n1 ${file} >> all_pdb_viol_summary.txt
done
```

      3. The format for the individual *cst_viol.txt files is as follows:

```
resA  atomA resB atomB experimental_value distance_in_model violation_magnitude
```

      4. At the end of the *cst_viol.txt file, there is a line that provides the total sum of all violations in that model (sum_current_viol), as well as the maximum distance violation for that model (max_current_viol).
      5. *OPTIONAL: If you have BOUNDED restraints, you can use the script called calc_exp_viol.pl in the same manner as above.*

Appendix: Additional Techniques

1. Using make_fragments.pl - (NOTE: The workshop machines are not configured for making fragments.)
    1. For more information on the fragment picker, please go to https://www.rosettacommons.org/docs/latest/app-fragment-picker.html and see Gront, et al., PLoS ONE, 2011.
    2. The make_fragments.pl script is in ~/rosetta_workshop/rosetta/tools/fragment_tools/
    3. Please read the fragments.README file in the above directory for more information concerning using the script
    4. In order to use make_fragments.pl, you will first need to install PSI-BLAST (ftp://ftp.ncbi.nih.gov/blast/executables/release/2.2.17/; Requires non-blast+ NCBI version), the non-redundant (NR) database (ftp://ftp.ncbi.nih.gov/blast/db/), and perhaps PSI-PRED (http://bioinf.cs.ucl.ac.uk/psipred/).
    5. You will also need a Vall database, which is in the above directory (vall.jul19.2011.gz)
    6. You will need to modify make_fragments.pl in order to reflect the paths specific to your case (we will not do this during the workshop). The comments in the scripts should tell you how to modify paths.*
    7. For a usage statement, run the script without arguments:

        ```
        ~/rosetta_workshop/rosetta/tools/fragment_tools/make_fragments.pl
        ```

    8. To run in the directory where your fasta file is located:

        ```
        ~/rosetta_workshop/rosetta/tools/fragment_tools/make_fragments.pl 2LZMA.fasta >& make_fragments.lo
        ```

    9. OPTIONAL: You can use any other secondary structure prediction method, such as JUFO (http://www.meilerlab.org/index.php/servers/show?s_id=5) or Porter (http://distill.ucd.ie/porter/), but it must be in PSI-PRED ss2 vertical format. This can be done by running:

        ```
        python2.7 ~/rosetta_workshop/rosetta/tools/fragment_tools/ss_pred_converter.py \
        -j 2LZMA.jufo_ss > 2LZMA.jufo_ss2
        ```

        or

        ```
        python2.7 ~/rosetta_workshop/rosetta/tools/fragment_tools/ss_pred_converter.py \
        -p 2LZMA.porter_ss > 2LZMA.porter_ss2
        ```

2. Using R (http://www.r-project.org) to generate score vs. rmsd plots.
    1. Open R by typing R on the command line.
    2. Read in the score table and put into a data object called data:

        ```
        data<-read.table("2LZM_rescore_vs_rmsd.table", header=TRUE)
        ```

    3. Look at the first line of data:

        ```
        data[1,]
        ```

    4. Plot score vs. rms_core_low:

        ```
        plot(data$rms_core_low,data$score,pch=4,cex.axis=1.5,xlab="",ylab="",main="")
        ```

    5. You may have to play with the range of the y-axis to see the distribution and excluding outliers.

        ```
        plot(data$rms_core_low,data$score,pch=4,cex.axis=1.5,xlab="",ylab="", \
        main="",ylim=range(min(data$score),-100))
        ```

    6. Next, add a title:

        ```
        title(main="2LZM_score_vs_rmsCoreLow",xlab="rms_core_low (Angstroms)", \
        ylab="score (REU)", cex.lab=1.5, cex.main=2.0)
        ```