Quick-start guide to ab-initio protein folding using Rosetta 3.1

This is one of a series of tutorials designed to get you started with protein structure prediction using Rosetta 3.1. It was produced to accompany Kaufmann et. al. (2010) Biochemistry, and the latest version can be found at http://meilerlab.org/. The purpose of ab initio folding is to predict the tertiary structure adopted by a protein, given only the primary sequence. The Rosetta ab-initio (de-novo) folding process uses statistics of the structures in the PDB to predict the conformation of short overlapping stretches (3-mers and 9-mers) of the sequence, combines them, and chooses the most energetically favorable structure. The algorithm is described in Simons et al. (1997) J Mol Biol 268, 209-225. It is assumed that you have installed the Rosetta suite from rosettacommons.org, and that you are comfortable working in Linux. You will also want to become familiar with the documentation that can be found in the /manual/ and /demos/ subdirectories, as well as the online Rosetta 3 User Manual, FAQ, and forums at http://www.rosettacommons.org/tiki/.

Steps to execute the folding run

1. You must provide the amino acid sequence of your protein of interest to Rosetta in fasta format (http://www.ncbi.nlm.nih.gov/blast/fasta.shtml). Rosetta has been benchmarked on small single-domain proteins, so it can be useful to perform domain prediction using online services such as Dompred, and then perform folding runs on separate domains up to ~200 aa in length. There is a sample sequence for the protein ubiquitin included with this tutorial: 1ubi.fasta (76 aa).
2. If the native structure of your protein is already known, you can optionally provide it for benchmarking purposes. There is a sample PDB file provided with this tutorial in the /denovo_folding/ directory: 1ubi.pdb, which has been simplified as much as possible using perl scripts found in the /BioTools/ directory of the Rosetta installation. This file has no header, only a single protein chain, and contains no solvent or heteroatoms.
3. Generate fragments files from which to build the protein structure. This can be done most easily using the Robetta server at http://robetta.bakerlab.org/. Making the fragments locally can also be done, by following the instructions found here:
   http://www.rosettacommons.org/manuals/rosetta3_user_guide/file_fragments.html
4. Create a flags file to specify the parameters Rosetta will use during this run, including the location and format of input and output files, and options that specify details of the algorithm. There is a sample flags file in the /denovo_folding/ directory included with this tutorial. It is worth reading the file in a text editor becuase it is well-commented and contains pointers to more documentation. The parameters of the run can be changed by editing this file- for example, increasing the number of structures to generate from 5 to 50K.
5. Conduct the folding run using the ab initio executable (you may have to specify the path to your executables directory) and your flags file, and capture the output into a logfile:

```
AbinitioRelax.linuxgccrelease @flags > folding.log
```

Analysis and notes

1. Examine the output files generated during this abinitio run. This includes the logfile, the 5 pdb structure files, a silent.out file containing multiple structures, and the scorefile called score.fsc. These are all plain text files, so you can read them with a pager or text editor. You should look in the user manual to learn the interpretation of the file formats.

2. Rosetta's output structures are often referred to as decoys. Load some of the decoys into your favorite molecular visualization program (Chimera, PyMOL, etc) and compare them with the native ubiquitin structure (1ubi.pdb). You should note that the models are produced by default without sidechains. These can be filled in by many tools, including a subsequent full-atom refine/relax step (q.v.) in Rosetta.
3. Alternatively, the folding run can be done using the -relax:fast flag, which combines that subsequent step at the cost of speed. Compare the included flags2 file which takes this approach. Re-run the command using **`@flags2`** and compare the results. To see the full set of options that are available, run the command use the -help flag alone.
4. It is actually redundant to output the decoys as pdbfiles, because the silent.out file holds multiple structures, encoded using backbone angles and the centroid or side-chain coordinates. It is also a plain text file, so you can examine your results in this compact form. You can then extract individual pdbfiles from the silent.out file by using the **`extract_pdbs.linuxgccrelease`** command, with flags to specify all or some of the structures.
5. The output structures can be ranked by the Rosetta energy score, where the lowest energy indicates the best structure. The Rosetta scores are located in the scorefile, but can also be found in the silent.out file, and at the end of each output PDB file.
6. Another method for analyzing the typically thousands of models produced by a Rosetta run is to cluster them by structural similarity, with the idea that the deepest (native) energy well is frequently the widest. Therefore the largest cluster should contain the most native-like structure. Rosetta can perform clustering analysis (see http://www.rosettacommons.org/manuals/rosetta3_user_guide/app_cluster.html) and the final selection of the best structure frequently involves a combination of clustering and energy score ranking.