De novo Folding in Rosetta 3.2

Rosetta Workshop
March 12, 2011
Vanderbilt University
Tutorial Outline

• What is *de novo* folding?
• AbinitioRelax Protocol
  – Necessary input files
  – Running AbinitioRelax
  – Extracting data and analyzing output
• Folding membrane proteins with MembraneAbinitio
  – How is it different from AbinitioRelax?
• Folding with restraints
• Useful references and websites
What is *De novo* Folding?

- We are folding *from the primary sequence* using secondary structure prediction and peptide fragments from the PDB
- Use the fragments to change the geometry of the protein and score to keep good fragment insertions

Figure courtesy of Jens Meiler
What can Rosetta3.2 Actually Fold?

T4-lysozyme C-terminal domain

- small, globular, soluble proteins

rhodopsin

- small, simple membrane proteins

...but not large, complex proteins

V-type Na\(^+\) ATP synthase subunit
Rosetta De novo Folding Protocol
Necessary Input Files for AbinitioRelax

- FASTA file of your protein sequence
- “Clean” PDB file of native structure (optional)
- Fragment library files
- Options file
What’s Happening When we Make Fragments?

- Vall database
- Primary sequence
- Secondary structure prediction
- NMR data (if applicable)
- Gather all possible fragments
- Score candidates based on input
- Keep the best $N$ fragment
  - default=200 per sequence
  - Write to fragment files

Figure courtesy of Dominik Gront
Setting Up Options for AbinitioRelax

• First, create a new file called 2LZM_abrlx.options

```bash
-in
  -file
    -native <native PDB file>       # native PDB file (optional)
    -fasta <primary sequence in FASTA format>  # protein sequence in fasta format
    -frag3 <3mer fragment file>    # protein 3-residue fragments file
    -frag9 <9mer fragment file>    # protein 9-residue fragments file
  -psipred_ss2 <PSIPRED secondary structure prediction file>  # psipred_ss2 secondary structure definition file (required for -use_filters)
  -abinitio
    -increase_cycles 10          # Increase the number of cycles at each stage in AbinitioRelax by this factor
    -rg_reweight 0.5            # Reweight contribution of radius of gyration to total score by this scale factor
    -rsd_wt_helix 0.5           # Reweight env, pair, and cb scores for helix residues by this factor
    -rsd_wt_loop 0.5            # Reweight env, pair, and cb scores for loop residues by this factor
    -relax                      # At the end of de novo folding, do a relax step
  -relax
    -fast                      # Type of relax protocol. This has been shown to be the best deal for speed and robustness.
  -use_filters true
```

# Use radius of gyration (RG), contact-order, and sheet filters. This option conserves computing by not continuing with refinement if a filter fails. A caveat is that for some sequences, a large percentage of models may fail a filter. The filters are meant to identify models with non-protein like features.
## Setting Up Options for AbinitioRelax Cont.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--run</td>
<td>-reinitialize_mover_for_each_job # Job distributor generates fresh copy of its mover before each apply (once per job)</td>
</tr>
<tr>
<td></td>
<td>-constant_seed # Use a constant seed (1111111 unless specified with -jran)</td>
</tr>
<tr>
<td></td>
<td>-jran 1111111 # Specify seed. Should be unique among jobs (requires -constant_seed)</td>
</tr>
<tr>
<td>-score</td>
<td>-find_neighbors_3dgrid # Use a 3D lookup table for doing neighbor calculations. For spherical, well-distributed conformations</td>
</tr>
<tr>
<td></td>
<td>-evaluation</td>
</tr>
<tr>
<td></td>
<td>-rmsd &lt;file to compute RMSD against&gt; &lt;column name&gt; &lt;file defining residues over which to compute RMSD&gt; # compute CA-RMSD for model comparing to native structure</td>
</tr>
<tr>
<td>-output</td>
<td># use this to tell Rosetta you actually want output</td>
</tr>
<tr>
<td></td>
<td>-nstruct 1 # how many structures do you want to generate? Minimum of 1000 recommended</td>
</tr>
<tr>
<td></td>
<td>-sf &lt;scorefile&gt; # full path to scorefile</td>
</tr>
<tr>
<td></td>
<td>-file</td>
</tr>
<tr>
<td></td>
<td>-silent &lt;silent output file&gt; # full path to silent file output</td>
</tr>
<tr>
<td></td>
<td>-silent_struct_type binary # we want binary silent files</td>
</tr>
<tr>
<td></td>
<td>-overwrite # overwrite any existing output with the same name you may have generated</td>
</tr>
</tbody>
</table>

**To run:**

```
$ROSETTA_BIN/AbinitioRelax.$ROSETTA_SUFFIX @2LZM_abrlx.options --database $ROSETTA_DATABASE &
2LZM_abrlx.log &
```
What’s Actually Happening?
Folding of Ubiquitin
Assessing Model Quality: Score vs. RMSD

• Determine how well Rosetta energy correlates with model quality (RMSD, MaxSub, etc.)
• If you don’t have a native structure (e.g., crystal structure) or a homolog that you’d like to compare the structure to, assume the lowest-scoring model is the native.
• Plot score vs. RMSD. Do you see “clusters” or populations of models? How does score relate to RMSD?
• Can also cluster (will be covered in another tutorial)
T4-Lysozyme Folding in Rosetta3.2: Compare to Lowest-Scoring
T4-lysozyme Folding in Rosetta3.2

![Graph showing score vs. CA-RMSD](image)
Looking at Models in PyMol

/Applications/MacPyMOL.app/Contents/MacOS/MacPyMOL
2LZM_.pdb S_00000175_3*.pdb S_00000129_1*.pdb
S_00000026_2*.pdb S_00000168_2*.pdb S_00000028_4*.pdb

Top 5 scoring

Best-scoring
Folding Membrane Proteins

- The steps to follow are basically the same (including making fragments) with a couple extra steps. Data analysis pretty much the same.
- RosettaMembrane divides up the membrane into hydrophobic, hydrophilic, and soluble layers.
- Membrane protein-specific scoring functions have been derived and are used in the MembraneAbinitio application.

Figure from Yarov-Yarovoy, Schonbrun, and Baker 2006.
Input Files

Spanfile - *.span
--transmembrane topology prediction file generated using octopus2span.pl script
--Input OCTOPUS topology file is generated at http://octopus.cbr.su.se/ using protein sequence as input.

Lipopholicity prediction file - *.lips4
--Generate using run_lips.pl script
--Need input FASTA file, spanfile, blaspgp and nr (NCBI) database to run

Fragment generation
-- Advised to use SAM but not JUFO or PSIPRED, which predict TMH regions poorly
Example Inputs and Command Line

**comment**

- TM region prediction for BRD4 predicted using OCTOPUS

**# TMHs**

- 4 123
- antiparallel
- n2c

**# residues**

**TM spans (2X)**

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>26</th>
<th>6</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>51</td>
<td>31</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>78</td>
<td>58</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>117</td>
<td>97</td>
<td>117</td>
<td></td>
</tr>
</tbody>
</table>

**spanfile**

Lipid exposed data:

<table>
<thead>
<tr>
<th>resnum</th>
<th>mean-lipo</th>
<th>lipophil</th>
<th>entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-1.000</td>
<td>3.004</td>
<td>1.211</td>
</tr>
<tr>
<td>9</td>
<td>-1.000</td>
<td>2.268</td>
<td>2.137</td>
</tr>
<tr>
<td>10</td>
<td>-1.000</td>
<td>4.862</td>
<td>1.095</td>
</tr>
<tr>
<td>13</td>
<td>-1.000</td>
<td>1.304</td>
<td>1.552</td>
</tr>
<tr>
<td>16</td>
<td>-1.000</td>
<td>3.328</td>
<td>2.025</td>
</tr>
</tbody>
</table>
MembraneAbinitio Options

-\texttt{--in:file:native <input native PDB>}
-\texttt{--in:file:fasta <primary sequence in FASTA format>}
-\texttt{--in:file:frag3 <3mer fragment file>}
-\texttt{--in:file:frag9 <9mer fragment file>}
-\texttt{--in:file:spanfile <spanfile> \# newly created spanfile}
-\texttt{--in:file:lipofile <lipophilicity lips4 file> \# newly created lipo file}

-\texttt{--run:reinitialize_mover_for_each_job}
-\texttt{--score:find_neighbors_3dgrid}
-\texttt{--abinitio:membrane \# specify membrane abinitio protocol}
-\texttt{--membrane:no_interpolate_Mpair \# membrane scoring specification}
-\texttt{--membrane:Menv_penalties \# turn on membrane penalty scores}
-\texttt{--rg_reweight 0.01 \# radius of gyration weight not so important for MPs}
-\texttt{--stage2_patch <score_membrane_s2.wts_patch> \# weights for scores}
-\texttt{--stage3a_patch <score_membrane_s3a.wts_patch> \# weights for scores}
-\texttt{--stage3b_patch <score_membrane_s3b.wts_patch> \# weights for scores}
-\texttt{--stage4_patch <score_membrane_s4.wts_patch> \# weights for scores}
-\texttt{--evaluation::gdtmm \# output global distance test info}

-\texttt{--out:nstruct 1 \# minimum of 1000 recommended}
-\texttt{--out:file:scorefile <path to scorefile>}
-\texttt{--out:file:silent <path to silent output file>}
-\texttt{--out:file:silent_struct_type binary}

\texttt{\$ROSETTA_BIN/membrane_abinitio2.\$ROSETTA SUFFIX @BRD4_mem_abrlx.options \--database \$ROSETTA_DATABASE >\& logfile &}
Folding with Restraints

- Basically the same as normal *de novo* folding protocol except add a few flags to options file
  - `fold_cst` # use FoldConstraints protocol
  - `force_minimize` # minimize in FoldConstraints protocol
  - `constraints`
    - `cst_file` ./2LZM_dist_w1.cst # path to your cst file
    - `cst_weight` 4 # factor by which total cst score multiplied by
    - `epr_distance` # Use RosettaEPR knowledge-based potential

<table>
<thead>
<tr>
<th>Constraint info</th>
<th>Constraint Function info</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cst_type&gt;</td>
<td>&lt;atom1&gt;</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
<tr>
<td>AtomPair</td>
<td>CB</td>
</tr>
</tbody>
</table>
What is with this Constraints File?

• There are constraint types and function types
  – Constraint types: AtomPair, Angle, Dihedral, etc.
  – Function types: Bounded, Spline, Harmonic, Gaussian, etc.

• Each constraint you define is scored individually, and the total constraint score is the sum of all individual scores

• Each constraint can have its own constraint type and function type.
  – In some cases, like when using Spline function, each constraint can have its own weight

• How you define the constraint and how it’s scored depends on the constraint type; this is same with function type.
Analysis After Folding with Restraints

• Can often filter by constraint score so that only look at models that satisfy experimental data the best
• Can plot score vs. RMSD, constraint score vs. RMSD, total score vs. constraint score, etc. to get idea of correlation of constraint score with total energy of model
• Can see how many violations your model has, how big the violations are, etc.

Figure courtesy of Nathan Alexander
A Few More Things to Keep in Mind

- Rosetta3.2 *de novo* folding performs best with small proteins (< 100 residues)
- Folding larger, more complex proteins probably requires more restraints
- *Can* fold membrane proteins with experimental restraints (EPR, NMR, etc.).
  - Exact protocol seems to depend on system and problem being addressed
- More folding capabilities in more recent versions that have not been released (more to come!)
Useful Links and Papers

• **Rosetta User’s Guide:**

• **De novo folding**

• **Membrane protein folding**

• **Using constraints/restraints in Rosetta 3.2**
  – [http://www.rosettacommons.org/manuals/archive/rosetta3.2_user_guide/constraints.html](http://www.rosettacommons.org/manuals/archive/rosetta3.2_user_guide/constraints.html)