Supplementary Information

Small-Molecule Ligand Docking into Comparative Models with Rosetta

Steven A. Combs[1,2,], Samuel L. DeLuca[1,3,*], Stephanie H. DeLuca[1,3,*], Gordon H. Lemmon[1,3,*], David P. Nannemann[1,2,*], Elizabeth D. Nguyen[1,3,*], Jordan R. Willis[1,3,*], Jonathan H. Sheehan[1,*], Jens Meiler[1,2,3,4,5†]*

Affiliations:
[1] Center for Structural Biology, Vanderbilt University, Nashville, TN, USA
[2] Department of Chemistry, Vanderbilt University, Nashville, TN, USA
[3] Chemical and Physical Biology Program, Vanderbilt University, Nashville, TN, USA
[4] Department of Pharmacology, Vanderbilt University, Nashville, TN, USA
[5] Department of Biomedical Informatics, Vanderbilt University, Nashville, TN, USA

[*]**These authors contributed equally.**

Authors' Email:
Steven A. Combs: steven.combs@vanderbilt.edu
Samuel L. DeLuca: samuel.l.deluca@vanderbilt.edu
Stephanie H. DeLuca: stephanie.h.deluca@vanderbilt.edu
David P. Nannemann: david.p.nannemann@vanderbilt.edu
Elizabeth D. Nguyen: e.dong.nguyen@vanderbilt.edu
Gordon H. Lemmon: gordon.h.lemmon@vanderbilt.edu
Jordan R. Willis: jordan.r.willis@vanderbilt.edu
Jonathan H. Sheehan: jonathan.sheehan@vanderbilt.edu

[†] **Correspondence should be addressed to:**
Jens Meiler
Email: jens.meiler@vanderbilt.edu
Telephone: +1 615 936 5662
Fax: +1 615 936 2211

Included:
Supplemental Discussion S1: Clustering using Rosetta. A description of how to perform clustering in Rosetta 3.4. Clustering can be performed using a variety of other programs (see text).
Supplemental Discussion S2: Using Constraints as Filters in RosettaScripts. Instructions on how to include experimental data as restraint filters during modeling with RosettaScripts. This may be easier than filtering models post hoc (see text).
Supplemental Method: Installing Rosetta 3.4. Instructions on how to install the current released version of Rosetta on Linux and Mac OS X operating systems.
Supplemental Discussion S3: Testing Rosetta. An example of how benchmarking of Rosetta is typically performed.

Nature Protocols: doi: 10.1038/nprot.2013.074

Supplemental Glossary: A list of terms and definitions commonly used in describing modeling with Rosetta.
Supplemental Tutorial: A zipped file that contains example input files, command lines, and output files that the user can download. This tutorial is provided to allow the user to gain hands-on experience with docking small molecules into comparative models with Rosetta.

Supplemental Methods

Installing Rosetta 3.4

The Rosetta modeling suite is free of cost to all academic users after registration. The package comes with a user's guide; a database containing pertinent files for applications in Rosetta; the Rosetta source code, and a fragments directory containing a peptide fragment database of proteins for known structures. In addition, the modeling suite comes with FoldIt[3], an interactive graphical interface that manually folds proteins using the Rosetta scoring function and structure prediction algorithm. ProteinTools (`rosetta_tools/protein_tools`), a collection of ancillary tools commonly used in conjunction with the Rosetta software suite, is also included. Lastly, Rosetta comes with a software construction tool called SCons (http://www.scons.org/), which analyzes the source code and builds specified binary files using multiple processors. SCons interfaces with the standard GNU gcc compiler to build the source code.

1. **Installing Rosetta version 3.4**

   (i) Rosetta is free to academic users. For an academic license, apply here: http://depts.washington.edu/uwc4c/express-licenses/assets/rosetta+pyrosetta/
   (ii) After obtaining a username and password, source code for Rosetta 3.4 can be downloaded here: https://www.rosettacommons.org/software/academic or https://www.rosettacommons.org/software/commercial
   (iii) Create a new directory called `rosetta` (`mkdir rosetta`), and copy `rosetta3.4_bundles.tgz` from your downloads directory to `rosetta/` (`cp rosetta3.4_bundles.tgz rosetta/`)
   (iv) In the `rosetta` directory, unpack the `tar` file by typing the following: `tar -zxvf rosetta3.4_bundles.tgz`. This will unpack multiple files, including the Rosetta source code (`rosetta3.4_source.tgz`) and the Rosetta database (`rosetta3.4_database.tgz`).
   (v) Unpack the database and source code using the following:

   ```
   tar -zxvf rosetta3.4_database.tgz
   tar -zxvf rosetta3.4_source.tgz
   ```

   (vi) Change directories into the newly created `rosetta3.4_source` (`cd rosetta3.4_source`), and build the binaries using the following command:

   ```
   external/scons-local/scons.py mode=release bin/
   ```

   CRITICAL: Make sure you have the GNU gcc3.4 compiler or higher (by typing `gcc -v`) and that a working copy of Python2.5 or higher installed. Access a Python executable by `/usr/bin/env python` or explicitly type the path to your python executable. Additionally, you must have zlib installed, if you see errors referencing a missing –lz library, install the zlib-dev libraries for your operating system.

   Note for Mac OS X Users: the compiler "clang" is recommended for compiling Rosetta in OS X. Make sure you have the OS X Developer tools installed, and

then compile using the following build command: `external/scons-local/scons.py cxx=clang mode=release bin/`

If an error similar to `KeyError: "Unknown version number 4.1 for compiler 'clang'"` occurs, open the file `rosetta_source/tools/build/options.settings` and modify the line:
`"clang" : [ "1.7", "2.1", "2.0", "2.8", "2.9", "3.0", "3.1","*" ],`

To instead read:
`"clang" : [ "1.7", "2.1", "2.0", "2.8", "2.9", "3.0", "3.1","4.1","*" ],`

Executables compiled in this way will have the suffix ".macosclangrelease"

(vii) Change directory into `bin/ (cd bin)`, and confirm that all 166 binaries have been built.

The Rosetta modeling suite comes with a set of python scripts, which can greatly simplify the analysis of Rosetta models. These scripts are used throughout this tutorial. The scripts rely on several python dependencies, which need to be installed. BioPython and numpy, which are freely available python packages, are required to use these scripts. The specific installation instructions for BioPython (http://biopython.org/wiki/Biopython) and numpy (http://numpy.scipy.org/) will vary based on your specific operating system details. Consult the documentation for these packages for installation instructions.

Rosettautil is a python module with a number of useful functions for handling Rosetta output. The installation package for this Python module is located in `<rosetta_dir>/rosetta_tools/protein_tools`. To install this module, and the associated scripts, first change directories to the `<rosetta_dir>/rosetta_tools/protein_tools` directory. If you have root access and want to install the module so that it is usable by all users on the system, enter the following command: `python setup.py --install-scripts=/path/to/scripts/directory`. This command will install the python module, and then copy the scripts into the directory you specify. If you do not have root access, use this command: `python setup.py --user --install-scripts=/path/to/scripts/directory.`

Supplemental Discussion

S1: Clustering Using Rosetta (OPTIONAL)

Predicted structures generated by comparative modeling are often clustered to help identify structurally similar models. Clustering is performed with the assumption that the deepest energy well, and hence the global energy minimum, will also be the widest[2]. As a result, it is expected that the largest clusters will potentially contain the predicted model that is closest to the native structure. Rosetta includes a tool for clustering protein models. The cluster application avoids the memory requirements associated with computing a complete distance matrix for large numbers of models. The Rosetta clustering method starts by computing a distance matrix for the first 400 input models. Each model in the distance matrix is assigned to the cluster to which it is nearest (typically in terms of RMSD). If the model is not within a specified radius of any cluster, it is assigned to a new cluster.

Because the Rosetta clustering application outputs most of its statistical information in its log file, a script has been provided to run the clustering application and produce a clear summary of the results. Given a set of PDB or Rosetta silent files and a Rosetta options file, `clustering.py` will produce a set of clustered PDBs, a histogram file showing the distribution of pairwise RMSDs between models, and a summary file showing which models are in which clusters. The Rosetta options file can contain a number of options that control the behavior of the cluster application. The acceptable options are listed below. For more information on Rosetta options file formatting, see the *Experimental design* section in the main text.

- `-run:shuffle` - Input structures in a random order. Use this if you have reason to believe that the output models are not in random order
- `-cluster:gdtmm` - cluster using GDTMM (global distance test) instead of RMSD distances
- `-cluster:radius <float>` - The maximum radius of each cluster in Å (RMSD mode) or inverse GDT_TS (GDTMM mode)
- `-cluster:input_score_filter <float>` - do not cluster scores above a given energy
- `-cluster:exclude_res <int> <int> ….<int>` - do not include the listed residues in distance calculation
- `-cluster:limit_cluster_size <int>` - maximum number of models in each cluster
- `-cluster:limit_clusters <int>` - maximum number of clusters
- `-cluster:limit_total_structures <int>` - maximum number of models to cluster
- `-cluster:sort_groups_by_energy` - sort clusters by total energy during output

S2: Using Constraints as Filters in RosettaScripts (OPTIONAL)

Filters can be used to guide a RosettaScripts[1] protocol in producing only high quality models that pass constraints specified by the user. These filters can take on a variety of forms, where the protocol can be repeated until a certain score is met (filter by score), continuing the protocol until the model converges on another structure (filter by RMSD), or continuing the protocol until atomic contacts are made which agree with experimental observations (filter by experimental constraints). For example, an atomic contact or residue contact constraint is essentially a distance constraint that can be derived from several types of experimental data, such as NMR nuclear Overhauser effects (NOEs), distances determined by electron paramagnetic resonance (EPR), distances derived from cysteine mutagenesis, etc. The protocol repeats for all or specified movers until the experimental constraints are satisfied. In the XML script used by RosettaScripts, all filters are specified in the `FILTERS` section as shown below:

```
<FILTERS>
	<filter1/>
	<filter2/>
	...
</FILTERS>
```

Then, in the `PROTOCOLS` section:

```
<PROTOCOLS>
	...
	<Add mover_name=mover1 filter_name=filter1>
	<Add filter_name=filter2/>
```

```
        ...
</PROTOCOLS>
```

The general format for a filter placed in the XML script is:

```
<"filter_name" name="your_filter_name" ..parameter_name=<parameter_value>,
```

where "`filter_name`" is one of a predefined set of filters recognized by RosettaScripts, and `name` is a unique identifier for this particular filter, followed by the parameters that the specific filter needs to be defined. In the example, `mover1` would continue until the constraints given in `filter1` are satisfied. For `filter2` that is not specified with a mover, the entire protocol would repeat up to `filter2`, until the constraints defined by the filter are met.

An example of an AtomicContact filter is:

```
<AtomicContact name="res32_res45_noe" residue1=32 residue2=45 sidechain=0
backbone=1 protons=0 distance=10 confidence=0.25/>
```

This filter will check if the model generated by Rosetta satisfies the specified constraint between residues 32 and 45. If any pair of backbone residues between the two residues is within 10Å of each other, the filter will return TRUE 75% of the time. When the filter returns a TRUE value, the protocol is continued from where the filter is called. In addition, the "`sidechain`" and "`protons`" options have designated values of 0 (as opposed to 1), which means they are turned off. The side chain and hydrogen atoms are not taken into account, and only the main chain backbone atoms will be evaluated in this filter example.

If the confidence is 1.0 then the filter is evaluated as either true or false. When the confidence value is less than 0.999, the filter will return TRUE in (1.0 – confidence) fraction of the times it is evaluated. This so-called "fuzzy" filter is useful for instances of ambiguous or uncertain experimental data.

A less sophisticated type of constraint filter conceptually similar to the AtomicContact constraint filter is the ResidueDistance filter. This filter queries the distance between the beta-carbons of two specified residues. An example similar to the AtomicContact filter is:

```
<ResidueDistance name="res32_res45_noe" res1_res_num=32 res2_res_num=45
distance=10.0 confidence=0.25/>
```

Finally, a DisulfideFilter can be applied, in which Rosetta tries to select models that position the specified residues such that they can potentially form a disulfide bond:

```
<DisulfideFilter name=disulfide targets=32,45,46 confidence=1.0/>
```

Notice that "`targets`" is defined as a comma-separated list of residues, which means that all numbers in the comma-separated list are considered when searching for a disulfide bond. For more information on using constraints in RosettaScripts[1], including constraint-type filters, go to: http://www.rosettacommons.org/manuals/archive/rosetta3.4_user_guide/RosettaScripts.html

S3: Testing Rosetta

Rosetta is often used to recapitulate known experiments. In such studies, several Rosetta options or protocol steps are changed carefully and methodically until the computational and experimental results correlate. Before conducting experiments with Rosetta, it is advised to test, or benchmark, the proposed protocol on known experimental data.

The definition of a successful benchmark varies and depends on the protocol. For a loop modeling benchmark, if the Rosetta model has a sub-angstrom RMSD to the experimental structure and is in the top ten lowest-scoring models built, the benchmark is considered to be successful[4]. For ligand docking, success is achieved when one of the top ten scoring models has an RMSD to the native structure below 2Å[5].

The data obtained from loop building for the T4-lysozyme comparative model (see the main text) can be used as an example of benchmarking. The RMSD from the native ligand position can be calculated across all generated models via the scripts provided with a copy of Rosetta. To calculate the RMSD, run the provided script:

```
scripts/score_vs_rmsd_singleproc.py --native=target_A.pdb --
table=score_vs_rmsd_loops_ca.txt --term=total --ca_mode=ca --chain=A
loops_final_*.pdb --res=residues.ls
```

For more information on calculating RMSDs to a native structure, see the main text (**STEP 14-16**, **Figure 4**) Additionally, RMSDs for the ligand-docked binding poses can be computed (**STEP 22**, **Figures 5**).

NOTE: The `InterfaceScoreCalculator`'s "`native`" option is used for benchmarking (if the native structure is known). To demonstrate the accuracy of the protocol in this example, the native structure is included in the script as `2ou0_native.pdb`. However, when running on a system where the native structure is not available, this option should be omitted. For reference, the results from this example will be compared against the experimentally determined T4-lysozyme crystal structure (referred to below as `2ou0_native.pdb`), which must also be aligned in the same coordinate system as the homology models used for docking. To enable RMSD calculation to the native, modify the `*.xml` script to include the native option:

```
<InterfaceScoreCalculator name=add_scores chains=X scorefxn=hard_rep
native="2ou0_native.pdb"/>
```

In your resulting `score.sc` file, the `interface_delta_X` is the score, and `ligand_rms_no_super_X` is the RMSD. NOTE: The RMSDs for the lowest-energy ligand docking models for this example are considered relatively large; however, this is not unexpected due to the small size and the symmetry of the ligand. The binding conformation and position of this low-energy cluster of models could be another energy minimum separate from that found in the crystal structure. Further, Rosetta is able to sample the native ligand conformation and position in a slightly higher-energy minimum. If the method you use is not yielding satisfactory test results, the size of the binding pocket search space can be decreased. For example, for this ligand (1-methyl-1H-pyrrole (MR3), a search radius of 2Å was used. In cases where the ligand is small and rigid, it is

better to limit the degrees of freedom in which the ligand moves around the binding pocket. However, for larger ligands, a radius of up to 4-5Å may be needed to accommodate the ligand.

Supplement Glossary

all-atom - in the case of sampling, synonymous with fine movements and often including side chain information; also referred to as high-resolution

benchmark – another word for a test of a method, scoring function, algorithm, etc. by comparing results from the method to accepted methods/models

binary file – a file in machine-readable language that can be executed to do something *in silico*

BioPython – a set of tools for biological computing written and compatible with Python http://biopython.org/wiki/Biopython

build – to compile the source code so it can be used as a program

centroid – in Rosetta centroid mode, side chains are represented as unified spheres centered at the residue's center of mass

cluster center – the geometric center of a cluster, or group, of models

clustering – in this case, grouping models with similar structure together

comparative model – a protein model where the primary sequence from one protein (target) is placed, or threaded, onto the three dimensional coordinates of a protein of known structure (template)language (binary)

cyclic coordinate descent (CCD) – based on robotics, CCD loop closure is used to build loops in Rosetta by fragment assembly and close loops by decreasing the gap between two termini in three-dimensional space[6]

de novo – in this case, from the sequence; also called *ab initio*

directory – synonymous with a folder, usually contains one or more files or other folders

distance matrix – a matrix containing the pairwise distances for every point in a set of points

Dunbrack rotamer library – a set of likely side chain conformations for the twenty canonical amino acids based on protein structures in the Protein Data Bank (PDB)[7]

executable – binary file used to execute the program

force field / scoring function / energy function / potential – often used interchangeably; a means of assessing the energy of the generated models

fragment – in Rosetta folding and loop building, a set of three-dimensional coordinates corresponding to a given amino acid sequence

fragment database – also called the fragment library; contains the Cartesian coordinates for 200 amino acid fragments (obtained from the Vall database) for each sequence window of the entire primary sequence of the protein

gap – in sequence alignment, a gap is inserted when the sequences are of low homology; usually appear as a dash (-); the gaps form a sequence alignment correspond to areas where loops are built during comparative modeling

GDT / GDT_TS – global distance test (total score); a measure of similarity between two protein structures having the same amino acid sequence; the largest set of residues' Cα atoms in the model structure falling within a defined distance cutoff of their position in the experimental structure

gradient-based minimization – also known as minimization by steepest descent; in this case, a means of energy minimization in which one takes steps proportional to the negative of the gradient of the function (energy) at the current point

high-resolution – in the case of sampling, synonymous with fine movements and often including side chain information

homology model – a more specific type of comparative model where the protein sequence of interest (target) is a homolog of the protein of known structure (template)

interface delta - The interface delta score is defined as the contribution to the total score for which the presence of the ligand is responsible.

kinematic loop closure (KIC) – robotics-inspired loop closure algorithm which analytically determines all mechanically accessible conformations for torsion angles of a peptide chain using polynomial resultants[4]

knowledge-based – in the case of Rosetta, based on information obtained from structures found in the PDB

libraries – in computing, a collection of code and data (classes and functions) used by a piece of software and is often used in software development

ligand – in this case, a small molecule that binds to a protein to serve some biological purpose; in the presented protocol, the ligand (small molecule) is docked into the receptor (protein).

low-resolution – a somewhat subjective term, in the case of sampling, synonymous with coarse movements of the protein and/or ligand backbone and side chains; the individual atoms of low-resolution structures or models cannot be resolved, or observed.

metropolis criterion – often combined with the monte carlo sampling algorithm; allows for generation of an ensemble that represents a probability distribution; see Metropolis, *et al*., 1953[8]

model – in the case of this protocol, a structure generated by Rosetta; sometimes called a decoy

monte carlo sampling – a randomized and repetitive computational sampling method

mover - a generic class that takes as input a pose and performs some modification on that pose; for example, a mover might take in a pose and rotate every residue

namespace – in computer science, an abstract container holding a logical grouping of unique identifiers or symbols; in Rosetta, examples of namespaces are `loops`, `relax`, etc.

native-like – close to the experimentally determined structure; a model that is "native-like" usually has an RMSD to the experimentally determined structure of < 2Å.

options file – often called a flags file; a file containing Rosetta options that can be passed to a Rosetta executable after the @ symbol; can be easier to use than passing Rosetta options over the command line

pack / repack – in Rosetta, side chains are packed/repacked by switching out rotamers and scoring them using the Rosetta scoring function

params file – ligand file; defines the Rosetta atom-typing and internal coordinates of a small molecule; in the manuscript, specified by *.params

path – in this case, the location in the file system of a file or directory

physics-based – in the case of scoring functions, based on Newtonian physics; for example, two atoms are considered to be balls connected by a spring; often used in molecular mechanics

pose – in this Rosetta protocol, a three-dimensional conformation of the ligand, protein, or ligand/protein complex

Python – interpreted, object-oriented, high-level programming language http://www.python.org/

relax – in Rosetta, an iterative protocol of side chain repacking and gradient-based minimization; often referred to as full-atom (or all-atom) refinement

Robetta – Rosetta structure prediction server http://robetta.bakerlab.org/ freely available to not-for-profit users

RosettaCommons – a group of more than twenty labs that develop the Rosetta software suite

Rosetta energy units (REU) – *not* experimental energy units; these are arbitrary energy units specific to the Rosetta scoring function

RosettaScripts – also called the Scripter or RosettaXML; an XML-like language that allows for specifying modeling tasks in Rosetta[1]

rotamer – rotational conformer of an amino acid or ligand side chain

SCons – a tool for constructing software from its source code http://www.scons.org/

script - in computer programming, a script is a sequence of instructions that is interpreted or carried out by another program rather than by the computer processor (as a compiled program is)

source code – human-readable files that are the implementation of the program; in Rosetta, these are written in C++.

target – in comparative, or homology, modeling, the protein for which we are generating a model; the target sequence is the primary sequence of the protein for which we want to make a model.

template – in comparative modeling, the protein of known structure on which the target is threaded

threading – placing the primary sequence of one protein (target) on the three-dimensional coordinates of a protein of known structure (template) based on a sequence alignment

Vall database – database from which amino acid fragments are picked for folding and CCD loop building in Rosetta

XML – Extensible Markup Language; in this case, used to write protocols to pass to RosettaScripts

Supplemental Tutorial

The Supplemental Tutorial is available for download on the *Nature Protocols* website.

Supplemental References

1. Fleishman, S. J. *et al.* RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. *PLoS ONE* **6**, e20161 (2011).
2. Shortle, D., Simons, K. & Baker, D. Clustering of low-energy conformations near the native structures of small proteins. *Proc Natl Acad Sci U S A* **95**, 11158–11162 (1998).
3. Callaway, E. The shape of protein structures to come. *Nature* **449**, 765 (2007).
4. Mandell, D. J., Coutsias, E. A. & Kortemme, T. Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling. *Nat Meth* **6**, 551–552 (2009).
5. Davis, I. W. & Baker, D. RosettaLigand docking with full ligand and receptor flexibility. *J Mol Biol* **385**, 381–392 (2009).
6. Canutescu, A. & Dunbrack, R., Jr Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Sci* **12**, 963 (2003).
7. Dunbrack, R. L. & Karplus, M. Backbone-dependent rotamer library for proteins. Application to side-chain prediction. *J Mol Biol* **230**, 543–574 (1993).
8. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21**, 1087 (1953).

Nature Protocols: doi: 10.1038/nprot.2013.074