BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 5 2010, pages 689–691
doi:10.1093/bioinformatics/btq007

Structural bioinformatics Advance Access publication January 7, 2010

PyRosetta: a script-based interface for implementing molecular modeling algorithms using Rosetta

Sidhartha Chaudhury[1], Sergey Lyskov[2] and Jeffrey J. Gray[1,2,3,*]

[1]Program in Molecular Biophysics, [2]Department of Chemical and Biomolecular Engineering and [3]Institute of NanoBioTechnology, Sidney Kimmel Comprehensive Cancer Center and Institute of Computational Medicine, Johns Hopkins University, 3400 North Charles Street, Baltimore, MD 21218, USA

Associate Editor: Anna Tramontano

ABSTRACT

Summary: PyRosetta is a stand-alone Python-based implementation of the Rosetta molecular modeling package that allows users to write custom structure prediction and design algorithms using the major Rosetta sampling and scoring functions. PyRosetta contains Python bindings to libraries that define Rosetta functions including those for accessing and manipulating protein structure, calculating energies and running Monte Carlo-based simulations. PyRosetta can be used in two ways: (i) interactively, using iPython and (ii) script-based, using Python scripting. Interactive mode contains a number of help features and is ideal for beginners while script-mode is best suited for algorithm development. PyRosetta has similar computational performance to Rosetta, can be easily scaled up for cluster applications and has been implemented for algorithms demonstrating protein docking, protein folding, loop modeling and design.

Availability: PyRosetta is a stand-alone package available at http://www.pyrosetta.org under the Rosetta license which is free for academic and non-profit users. A tutorial, user's manual and sample scripts demonstrating usage are also available on the web site.

Contact: pyrosetta@graylab.jhu.edu

Received on November 12, 2009; revised on January 4, 2010; accepted on January 5, 2010

1 INTRODUCTION

Recent advances in molecular modeling have lead to its increasing use in structural biology research for a wide range of applications. The Rosetta biomolecular modeling suite, in particular, has proved effective in many diverse tasks including *ab initio* structure prediction and homology modeling (Raman *et al.*, 2009), protein and small-molecule docking (Chaudhury and Gray, 2007; Davis and Baker, 2009), loop modeling (Mandell *et al.*, 2009) and design (Kuhlman *et al.*, 2003). To make these protocols more accessible, a number of web-based servers have been constructed, such as Robetta (Chivian *et al.*, 2004), RosettaDock (Lyskov and Gray, 2008) and RosettaAntibody (Sivasubramanian *et al.*, 2008). However, many modeling problems do not fit cleanly into one of the standard Rosetta protocols, and algorithms that combine elements from different methods within Rosetta are often required to adequately model a particular system. Developing such algorithms requires extensive experience in both C++ programming and Rosetta software development, severely limiting its accessibility.

To make custom molecular modeling using Rosetta accessible to a broader community of structural biologists, we developed PyRosetta, a Python-based implementation of the Rosetta molecular modeling suite. Our goal was to enable users to define a molecular modeling problem, design an algorithm to solve it and implement that algorithm on the computer using preexisting Rosetta objects and functions. PyRosetta takes advantage of the object-oriented architecture of the new Rosetta release v3.1 to provide users with easy access to all the major functions and objects used by Rosetta developers (Leaver-Fay,A. *et al.*, manuscript in preparation). PyRosetta can be run in two modes: interactive-mode, which contains tab-completion and help features which are ideal for beginners, and script-mode, which is better suited for algorithm development. We chose Python as the scripting language because it is a sophisticated programming language that enjoys widespread use in the biology community and allows PyRosetta to be compatible with other Python-based packages such as PyMol (DeLano, 2002) and Bio-Python (Cock *et al.*, 2009). Our hope is that the extensive online communities of users of the many Python-based bioinformatics tools will help develop and share interfaces with PyRosetta. Since familiarity with Rosetta objects and functions is essential for new users, a tutorial, user's manual and sample scripts demonstrating usage are available on the web site.

We used a number of tools to convert the classes and functions in the Rosetta C++ source code into a Python-accessible form. GCC-XML (Kitware Inc., 2007) parses the classes and functions of the Rosetta C++ code into an XML representation using the GCC compiler. The Py++ package (Language Binding Project, 2009) uses the GCC-XML objects and generates Python bindings using the Boost.Python library (Boost, 2009). To make this process feasible for over 2000 Rosetta objects, this entire process is automated. The scripts are portable and tested on Mac OSX, Linux and Windows platforms. The building process requires 4–6 h depending on the platform and the pregenerated binary libraries are provided for download for all three platforms. A version of PyRosetta will be made available with each new release of Rosetta along with intermediate versions that add additional features, fix bugs, improve accessibility or expand documentation. In terms of computational cost, PyRosetta performs almost identically to the C++ build of Rosetta with performance benchmarks indicating a $<5\%$ difference in speed.

*To whom correspondence should be addressed.

2 ROSETTA APPROACH

Molecular modeling in Rosetta for structure prediction and design relies on the thermodynamic principle that the configuration of a biomolecular system at equilibrium tends toward that which is the lowest in free energy. The free energy of a given configuration (structure and sequence) is approximated using a score function that uses mathematical models of the major biophysical forces (Van der Waals energies, hydrogen bonding, electrostatics, solvation energies etc.) as a function of the configuration. Since it is impossible to exhaustively sample the entire configurational space accessible to the system because of its size and complexity, the starting structures and sampling strategies vary across different modeling applications. Furthermore, different energy scoring components carry different degrees of importance in different modeling applications. The necessity of tailored sampling and scoring strategies underscores the need for a generalized approach to implementing custom molecular modeling algorithms.

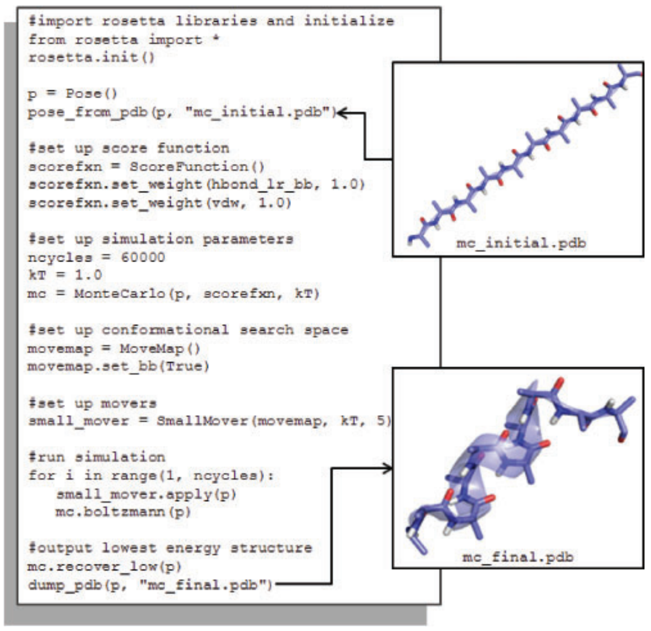
Rosetta protocols generally sample the relevant configurational space for a given modeling application by running a large number of relatively short Monte Carlo trajectories starting from random or semi-random starting configurations, storing the lowest energy structures from each trajectory (called decoys), and then selecting lowest energy decoys as predictions. To tackle a wide range of biomolecular modeling problems, it is necessary to precisely define the relevant configurational space for sampling, the search strategy and the score function used for both sampling and identifying the lowest energy structures.

3 PYROSETTA FEATURES

In PyRosetta, a biomolecular system is represented by an object called the `Pose`. The `Pose` contains all the structure and sequence information necessary to completely define the system in both Cartesian and internal coordinates. The internal coordinates used for proteins include the backbone torsion angles, ϕ, ψ and ω; the side chain torsion angles χ; and 'jumps' which define the relative position and orientation of multiple continuous polypeptide chains in the system. Nucleic acids and other molecules (ligands, post-translational modifications, etc.) are similarly built using internal coordinates; solvent is typically treated implicitly. The spatial and internal coordinates are synchronized within the `Pose`. A starting structure is read into a `Pose` from a Protein Data Bank file and its conformation is altered by perturbing its internal coordinates (Fig. 1).

The energy of a structure, or `Pose`, can be calculated using the `ScoreFunction`. The `ScoreFunction` represents an energy function that is the sum of weighted independent energy terms. Over 20 energy terms are available including Van der Waals attractive and repulsive components, hydrogen bonding, solvation and electrostatics energies. The user can create a custom scoring function by setting the weights of the desired components to non-zero values (Fig. 1). The energy of a structure is calculated by passing a pose into the scoring function, which returns the score.

Sampling functions are written as `Mover` objects in Rosetta. As a general form, a `Mover.apply(pose)` function carries out that particular perturbation on that `Pose`. Examples of movers include those that perturb backbone torsion angles, such as `SmallMover`, or minimize the structure using a given energy function, such as a `MinMover`. The conformational search space can be limited and controlled using the `MoveMap` and `FoldTree` objects. The `MoveMap` specifies which internal coordinates are to be held rigid during sampling. The `FoldTree` instructs the `Pose` on how to convert its internal coordinates into Cartesian coordinates (Bradley and Baker, 2006) and provides a framework for controlling how local perturbations propagate through the global structure (e.g. in loop modeling). Finally, the `MonteCarlo` object performs the bookkeeping for a Monte Carlo simulation, including storing the current structure and energy, calculating the change in energy caused by a perturbation and applying the Metropolis Criterion using the `MonteCarlo.boltzmann(Pose)` function (Fig. 1).



```
#import rosetta libraries and initialize
from rosetta import *
rosetta.init()

p = Pose()
pose_from_pdb(p, "mc_initial.pdb")

#set up score function
scorefxn = ScoreFunction()
scorefxn.set_weight(hbond_lr_bb, 1.0)
scorefxn.set_weight(vdw, 1.0)

#set up simulation parameters
ncycles = 60000
kT = 1.0
mc = MonteCarlo(p, scorefxn, kT)

#set up conformational search space
movemap = MoveMap()
movemap.set_bb(True)

#set up movers
small_mover = SmallMover(movemap, kT, 5)

#run simulation
for i in range(1, ncycles):
   small_mover.apply(p)
   mc.boltzmann(p)

#output lowest energy structure
mc.recover_low(p)
dump_pdb(p, "mc_final.pdb")
```

Fig. 1. A Monte Carlo peptide folding simulation using PyRosetta.

In addition to the basic movers, there are movers that execute standard Rosetta protocols. Examples include `DockingLowRes()` for the low-resolution phase of protein docking, `LoopMover_RefineCCD`, for the high-resolution refinement phase of loop modeling, and the versatile `PackRotamersMover`, which carries out side chain packing and design using a rotamer library. Finally, the `PyJobDistributor` object manages multiple simulation trajectories simultaneously while storing all the decoys and tabulating a score file. Scripts demonstrating molecular modeling protocols such as alanine scanning (Kortemme and Baker, 2002), protein and small-molecule docking (Chaudhury and Gray, 2007; Davis and Baker, 2009), protein design (Kuhlman and Baker, 2000) and all-atom relaxation (Bradley *et al.*, 2005) can be found on the PyRosetta web site.

4 OUTLOOK

PyRosetta is a Python-based implementation of the Rosetta molecular modeling package that enables users to implement molecular modeling algorithms using preexisting Rosetta objects and functions for a range of purposes from simple scripts to sophisticated modeling protocols and run them on the user's own computational resources. PyRosetta is stand-alone package requiring only Python 2.5 to be installed and is currently available for download from the web site (www.pyrosetta.org), along with a user's manual and sample scripts that demonstrate usage. For new users, we have written a set of interactive educational modules available both electronically and in a bound form (Gray *et al.*, 2009). The modules use PyRosetta to lead users from the fundamentals of biomolecular structure and energetics through algorithm creation for applications in structure prediction and design.

In the future, both Rosetta developers and outside users will be able to upload and share scripts with the PyRosetta community through the web site. The features described here are only a small subset of those available. Potential users are referred to the web site for more information.

ACKNOWLEDGEMENTS

We acknowledge William Sheffler for developing the first Python-bindings to Rosetta. John D. Bagert and Julian N. Rosenberg assisted in the early development through a Technology Fellowship from the Johns Hopkins University Center for Educational Resources. Finally, we acknowledge the efforts of all the Rosetta developers within RosettaCommons (www.rosettacommons.org) who have contributed the scientific research and software development to Rosetta that has made PyRosetta possible.

Funding: PyRosetta was funded through National Institute of Health (R01-GM73151 and R01-GM078221); National Science Foundation CAREER Grant CBET (0846324).

Conflict of Interest: none declared.

REFERENCES

Berrondo,M. *et al.* (2008) Structure prediction of domain insertion proteins from structures of individual domains. *Structure*, **16**, 513–527.

Boost (2009) Boost C++ libraries. Available at http://www.boost.org

Bradley,P. and Baker,D. (2006) Improved beta-protein structure prediction by multilevel optimization of nonlocal strand pairings and local backbone conformation. *Proteins*, **65**, 922–929.

Bradley,P. *et al.* (2005) Toward high-resolution de novo structure prediction for small proteins,.*Science*, **309**, 1868–1871.

Chaudhury,S. and Gray,J.J. (2008) Conformer selection and induced fit in flexible backbone protein-protein docking using computational and NMR ensembles. *J. Mol. Biol.*, **381**, 1068–1087.

Cock,P.J. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Davis,I.W. and Baker,D. (2009) RosettaLigand docking with full ligand and receptor flexibility. *J. Mol. Biol.*, **385**, 381–392.

DeLano,W.L. (2002) PyMOL molecular graphics system. Available at http://www.pymol.org/

Gray,J.J. *et al.* (2009) *The PyRosetta Interactive Platform for Protein Structure Prediction and Design: A Set of Educational Modules*.

Hecht,H.J. *et al.* (1992) Three-dimensional structure of a recombinant variant of human pancreatic secretory trypsin inhibitor (Kazal type). *J. Mol. Biol.*, **225**, 1095–1103.

Kim,D.E. *et al.* (2004) Protein structure prediction and analysis using the Robetta server, **32**, 526–531.

Kitware Inc. (2007) GCC-XML. Available at http://www.gccxml.org

Kortemme,T. and Baker,D. (2002) A simple physical model for binding energy hot spots in protein-protein complexes. *Proc. Natl Acad. Sci. USA*, **99**, 14116–14121.

Kuhlman,B. and Baker,D. (2000) Native protein sequences are close to optimal for their structures. *Proc. Natl Acad. Sci. USA*, **97**, 10383–10388.

Kuhlman,B. *et al.* (2003) Design of a novel globular protein fold with atomic-level accuracy. *Science*, **302**, 1364–1368.

Language Binding Project (2009) C/C++ Python language binding. Available at http://www.language-binding.net

Lyskov,S. and Gray,J.J. (2008) The RosettaDock server for local protein-protein docking. *Nucl. Acids Res.*, **36**, 233–238.

Mandell,D.J. *et al.* (2009) Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling. *Nat. Methods*, **6**, 551–552.

Raman,S. *et al.* (2009) Structure prediction for CASP8 with all-atom refinement using Rosetta. *Proteins*, **77**, 89–99.

Sivasubramanian,A. *et al.* (2008) High-resolution homology modeling of antibody Fv regions using knowledge-based techniques, de novo loop modeling and docking. *Proteins*, **74**, 497–514.