How To Make the Most of PyRosetta

Jason W. Labonte & Michael Pacella
Gray Lab

RosettaCON 2013

Outline

- How-to
  - `PyJobDistributor`
  - `MPI`
  - Passing & parsing Rosetta option flags
  - RNA
  - NMR constraints
  - Examine atom-pair energies
  - Symmetry operations
  - Non-canonical AAs & ligands
  - Custom movers and scoring terms
- Where else to go for help
  - Printing Rosetta objects
  - Demos & test scripts

PyRosetta Tutorial

HOW-TO

Job Distribution in PyRosetta: PyJobDistributor

```
jd = PyJobDistributor("filename", nstruct, sf)
# The above constructs a job distibutor that will create nstruct decoys
# named filename_1.pdb to filename_N.pdb and a score file, filename.fasc.
# The PyJobDistributor will not overwrite a file already in existence.
# When initialized, the next available output file is started as an in-
# progress file.

jd.native_pose = native_pose
# If a native pose is provided, a column of RMSDs will be included in the
# score file.

while not jd.job_complete:
    pose.assign(start_pose)
    my_protocol.apply(pose)
    jd.output_decoy(pose)
    # Outputs the next decoy, deletes the in-progress file, and creates the
    # next available in-progress file.
```

MPI

```
def my_function(decoy_num):
    filename = "decoy_" + str(decoy_num)

    my_protocol = MyClass()
    pose = my_protocol.run()

    pose.dump_pdb(filename)

rosetta.mpi_init()

rosetta.MPIJobDistributor(100, my_function)
```

Option Flags in PyRosetta

- `init(extra_options="-mute basic -mute core -mute protocols")`
- `init(extra_options="-include_surfaces -preserve_header true")`
- `init(extra_options="-include_sugars -read_pdb_link_records -enable_branching")`

This will add-on to a default list of options:

- `-database`
- `-ex1`
- `-ex2aro`

RNA in PyRosetta: To Do Beforehand

- pdb files with RNA must be in a special format to be imported into Rosetta.
  - Residue names GUA (G), ADE (A), CYT (C), & URA/URI (U) must be changed to rG, rA, rC, & rU, respectively, so that Rosetta knows they have ribose, not deoxyribose, rings.
  - A handy script in the `/toolbox` folder, `make_rna_rosetta_ready.py`, has been written to do this for you.

RNA in PyRosetta: Sample Code

```
# Create residue type set for RNA.
rna_set = ChemicalManager.get_instance().residue_type_set("rna").get()

# Load pose.
pose = pose_from_pdb(rna_set, "filename.pdb")

# RNA has different torsion angles....
print pose.gamma(1)  # 1 is the residue number.
print pose.delta(1)
print pose.epsilon(1)
print pose.chi(1)
print pose.zeta(1)

# Construct an RNA score function.
sf = create_score_function("rna_hires")
```

RNA in PyRosetta: Sample Code

```
# Import RNA movers and protocols.
from rosetta.protocols.rna import *

# Construct an RNA minimization mover.
min_mover = RNA_Minimizer()

# Minimize the pose.
min_mover.apply(pose)
```

NMR Constraints in PyRosetta: ConstraintSetMover

```
# Construct constraint set mover.
set_constraints = ConstraintSetMover()
set_constraints.constraint_file("filename.cst")

# Prepare scorefunction.
sf = get_fa_scorefxn()
sf.set_weight(atom_pair_constraint, 1.0)

# Set constraints into pose.
set_constraints.apply(pose)

# Score the pose.
sf.show(pose)
```

NMR Constraints in PyRosetta: List of Constraint Scoring Components

- `atom_pair_constraint`
- `constant_constraint`
- `coordinate_constraint`
- `angle_constraint`
- `dihedral_constraint`

Calculating Atom-Pair Energies

```
atom1 = residue_1.atom(i)
atom2 = residue_2.atom(j)
atr, rep ,solv = etable_atom_pair_energies(atom1, atom2,
sfxn)

print atr, rep, solv
```

Symmetry in PyRosetta: To Do Beforehand

- prepare a pdb of the "master" subunit
- prepare a symmetry definition file
- include `-symmetry:symmetry_definition name_of_symm_def_file.dat` in your `extra_options`

Symmetry in PyRosetta: Sample Code

```
# Extra import statements are necessary.
import rosetta.core.conformation.symmetry
import rosetta.core.pose.symmetry
import rosetta.core.scoring.symmetry
import rosetta.protocols.simple_moves.symmetry

# Create a symmetric pose.
def symmetrize_pose(pose):
    pose_symm_data = core.conformation.symmetry.SymmData(pose.n_residue(),
                                                         pose.num_jump())
    pose_symm_data.read_symmetry_data_from_file("sym_def_file.dat")
    core.pose.symmetry.make_symmetric_pose(pose, pose_symm_data)

    # Many other useful utility funtions are in core.pose.symmetry.
```

Symmetry in PyRosetta: Sample Code

```
# Create a symmetric scorefuction.
sym_sfxn = core.scoring.symmetry.SymmetricScoreFunction()

# Create a symmetric pack rotamers mover.
sym_packer = protocols.simple_moves.symmetry.SymPackRotamersMover(sym_sfxn,
                                                                  task)


# Create a symmetric min mover.
sym_min_mover = protocols.simple_moves.symmetry.SymMinMover()

# Create a symmetric move map.
move_map = MoveMap()
core.pose.symmetry.make_symmetric_movemap(pose, move_map)

# Many other useful movers are in protocols.simple_moves.symmetry.
```

Custom Parameter Files in PyRosetta
To Do Beforehand

- Obtain an `.mdl`-formatted file of your residue's geometry. (OpenBabel is great for converting formats on chemical structures.)
- Run `molfile_to_params.py` to convert to a Rosetta-readable `.params` file
- (Note, you may need to manually adjust some things in the file.)

Custom Parameter Files in PyRosetta: Sample Code

```
# Create a vector1 of paths to your extra .params files you want loaded.
params_paths = Vector1(["list", "of", "paths", "to", "extra", "params"])

# Create a non-standard ResidueTypeSet that includes your extra .params.
nonstandard_residue_set = generate_nonstandard_residue_set(params_paths)

# Use this ResidueTypeSet when loading your pdb w/ non-standard residues.
pose = pose_from_pdb(nonstandard_residue_set, "nonstandard.pdb")
```

Custom Parameter Files in PyRosetta: Another Option

- A more permanent route (though inappropriate for check-ins) is to add your new `.params` file to the chemical database.
- You will also need to specify the path in `residue_types.txt` (also in the database) and ensure it is not commented out.

PyRosetta Tutorial

WHERE TO GO FOR HELP

pyrosetta.org

Includes:

- A list of workshop tutorials in pdf format
- A few web tutorials
- Last year’s version of this presentation

Printing Objects in PyRosetta

- The Gray Lab has methodically been going through classes in the Rosetta library and adding print functionality.
- *E.g.*:

```
>>>min_mover = MinMover()
>>>print min_mover
Mover name: MinMover, Mover type: MinMover, Mover current
tag:NoTag
Minimization type: linmin, Score tolerance: 0.01, Nb list: 1,
Deriv check: 0
```

Demos & Test Scripts

- A large selection of demos can be found in your PyRosetta install directory in the `/demos` folder.
- Also visit the `/apps` directory for a (currently small) list of full applications.

Special Thanks

- Will Sheffler
- Jeff Gray
- Sid Chaudhury
- Sergey Lyskov
- Evan Baugh
- Boon Uranukul
- Gray Lab
- you, for listening