Customizing Rosetta protocols with RosettaScripts

Jacob McKinney

Meiler Lab

E-Mail: Jacob.Mckinney@vanderbilt.edu

Rosetta applications do not cover all protocols

ls /rosetta-3.13/main/source/bin/

```
ensemble_analysis.linuxgccrelease@
ensemble_generator_score12_sidechain_ver2.default.linuxgccdebug@
ensemble_generator_score12_sidechain_ver2.default.linuxgccrelease@
ensemble_generator_score12_sidechain_ver2.linuxgccdebug@
ensemble_generator_score12_sidechain_ver2.linuxgccrelease@
enzyme_design.default.linuxgccdebug@
enzyme_design.default.linuxgccrelease@
enzyme_design.linuxgccdebug@
enzyme_design.linuxgccrelease@
erraser2.default.linuxgccdebug@
erraser2.default.linuxgccrelease@
erraser2.linuxgccdebug@
erraser2.linuxgccrelease@
erraser_minimizer.default.linuxgccdebug@
erraser_minimizer.default.linuxgccrelease@
erraser_minimizer.linuxgccdebug@
erraser_minimizer.linuxgccrelease@
exposed_strand_finder.default.linuxgccdebug@
exposed_strand_finder.default.linuxgccrelease@
exposed_strand_finder.linuxgccdebug@
exposed_strand_finder.linuxgccrelease@
extract_atomtree_diffs.default.linuxgccdebug@
extract_atomtree_diffs.default.linuxgccrelease@
extract_atomtree_diffs.linuxgccdebug@
extract_atomtree_diffs.linuxgccrelease@
extract_motifs.default.linuxgccdebug@
extract_motifs.default.linuxgccrelease@
extract_motifs.linuxgccdebug@
extract_motifs.linuxgccrelease@
extract_pdbs.default.linuxgccdebug@
extract_pdbs.default.linuxgccrelease@
extract_pdbs.linuxgccdebug@
extract_pdbs.linuxgccrelease@
fast_clustering.default.linuxgccdebug@
fast_clustering.default.linuxgccrelease@
fast_clustering.linuxgccdebug@
fast_clustering.linuxgccrelease@
FiberDiffractionFreeSet.default.linuxgccdebug@
FiberDiffractionFreeSet.default.linuxgccrelease@
FiberDiffractionFreeSet.linuxgccdebug@
FiberDiffractionFreeSet.linuxgccrelease@
fix_alignment_to_match_pdb.default.linuxgccdebug@
fix_alignment_to_match_pdb.default.linuxgccrelease@
fix_alignment_to_match_pdb.linuxgccdebug@
fix_alignment_to_match_pdb.linuxgccrelease@
fixbb.default.linuxgccdebug@
fixbb.default.linuxgccrelease@
fixbb.linuxgccdebug@
fixbb.linuxgccrelease@
FlexPepDocking.default.linuxgccdebug@
FlexPepDocking.default.linuxgccrelease@
FlexPepDocking.linuxgccdebug@
FlexPepDocking.linuxgccrelease@
FloppyTail.default.linuxgccdebug@
oop_design.linuxgccrelease@
optE_parallel.default.linuxgccdebug@
optE_parallel.default.linuxgccrelease@
optE_parallel.linuxgccdebug@
optE_parallel.linuxgccrelease@
packing_angle.default.linuxgccdebug@
packing_angle.default.linuxgccrelease@
packing_angle.linuxgccdebug@
packing_angle.linuxgccrelease@
packstat.default.linuxgccdebug@
packstat.default.linuxgccrelease@
packstat.linuxgccdebug@
packstat.linuxgccrelease@
parse_rosetta_script.default.linuxgccdebug@
parse_rosetta_script.default.linuxgccrelease@
parse_rosetta_script.linuxgccdebug@
parse_rosetta_script.linuxgccrelease@
partial_thread.default.linuxgccdebug@
partial_thread.default.linuxgccrelease@
partial_thread.linuxgccdebug@
partial_thread.linuxgccrelease@
pepspec_anchor_dock.default.linuxgccdebug@
pepspec_anchor_dock.default.linuxgccrelease@
pepspec_anchor_dock.linuxgccdebug@
pepspec_anchor_dock.linuxgccrelease@
pepspec.default.linuxgccdebug@
pepspec.default.linuxgccrelease@
pepspec.linuxgccdebug@
pepspec.linuxgccrelease@
PeptideDeriver.default.linuxgccdebug@
PeptideDeriver.default.linuxgccrelease@
PeptideDeriver.linuxgccdebug@
PeptideDeriver.linuxgccrelease@
peptoid_design.default.linuxgccdebug@
peptoid_design.default.linuxgccrelease@
peptoid_design.linuxgccdebug@
peptoid_design.linuxgccrelease@
performance_benchmark.default.linuxgccdebug@
performance_benchmark.default.linuxgccrelease@
performance_benchmark.linuxgccdebug@
performance_benchmark.linuxgccrelease@
per_residue_energies.default.linuxgccdebug@
per_residue_energies.default.linuxgccrelease@
per_residue_energies.linuxgccdebug@
per_residue_energies.linuxgccrelease@
per_residue_solvent_exposure.default.linuxgccdebug@
per_residue_solvent_exposure.default.linuxgccrelease@
per_residue_solvent_exposure.linuxgccdebug@
per_residue_solvent_exposure.linuxgccrelease@
phosphorylation.default.linuxgccdebug@
phosphorylation.default.linuxgccrelease@
phosphorylation.linuxgccdebug@
phosphorylation.linuxgccrelease@
pH_protocol.default.linuxgccdebug@
swa_protein_main.linuxgccrelease@
swa_rna_main.default.linuxgccdebug@
swa_rna_main.default.linuxgccrelease@
swa_rna_main.linuxgccdebug@
swa_rna_main.linuxgccrelease@
swa_rna_util.default.linuxgccdebug@
swa_rna_util.default.linuxgccrelease@
swa_rna_util.linuxgccdebug@
swa_rna_util.linuxgccrelease@
SymDock.default.linuxgccdebug@
SymDock.default.linuxgccrelease@
SymDock.linuxgccdebug@
SymDock.linuxgccrelease@
tcrmodel.default.linuxgccdebug@
tcrmodel.default.linuxgccrelease@
tcrmodel.linuxgccdebug@
tcrmodel.linuxgccrelease@
template_features.default.linuxgccdebug@
template_features.default.linuxgccrelease@
template_features.linuxgccdebug@
template_features.linuxgccrelease@
thermal_sampler.default.linuxgccdebug@
thermal_sampler.default.linuxgccrelease@
thermal_sampler.linuxgccdebug@
thermal_sampler.linuxgccrelease@
theta_ligand.default.linuxgccdebug@
theta_ligand.default.linuxgccrelease@
theta_ligand.linuxgccdebug@
theta_ligand.linuxgccrelease@
torsional_potential_corrections.default.linuxgccdebug@
torsional_potential_corrections.default.linuxgccrelease@
torsional_potential_corrections.linuxgccdebug@
torsional_potential_corrections.linuxgccrelease@
UBQ_E2_thioester.default.linuxgccdebug@
UBQ_E2_thioester.default.linuxgccrelease@
UBQ_E2_thioester.linuxgccdebug@
UBQ_E2_thioester.linuxgccrelease@
UBQ_Gp_CYD-CYD.default.linuxgccdebug@
UBQ_Gp_CYD-CYD.default.linuxgccrelease@
UBQ_Gp_CYD-CYD.linuxgccdebug@
UBQ_Gp_CYD-CYD.linuxgccrelease@
UBQ_Gp_LYX-Cterm.default.linuxgccdebug@
UBQ_Gp_LYX-Cterm.default.linuxgccrelease@
UBQ_Gp_LYX-Cterm.linuxgccdebug@
UBQ_Gp_LYX-Cterm.linuxgccrelease@
UnfoldedStateEnergyCalculator.default.linuxgccdebug@
UnfoldedStateEnergyCalculator.default.linuxgccrelease@
UnfoldedStateEnergyCalculator.linuxgccdebug@
UnfoldedStateEnergyCalculator.linuxgccrelease@
validate_database.default.linuxgccdebug@
validate_database.default.linuxgccrelease@
validate_database.linuxgccdebug@
validate_database.linuxgccrelease@
validate_rosetta_script.default.linuxgccdebug@
```

Why use a protocol interface to Rosetta?

- Rosetta applications are specifically developed for a use case

- For a certain scientific task you might want to:
  - Modify an existing protocol
  - Combine two protocols
  - Make an entire novel protocol

How to Make Custom Protocols

- C++ - Directly modify the Rosetta source code
- PyRosetta – Python bindings for directly interacting with Rosetta functions (http://www.pyrosetta.org/)

- RosettaScripts – XML based interface for creating protocols

A Fair Warning

- Scientific:
  - New protocols have to be tested and show evidence that they fullfil their task (benchmarking)

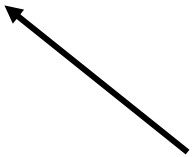
- Technical:
  - Not all XMLs/options have been tested in combination
  - Some XMLs require specific options to work

Running Rosetta Scripts

rosetta_scripts.linuxgccrelease -parser:protocol protocol.xml

The application

Runs whatever procedure is dictated in the XML file



The actual protocol

The file that describes your experimental steps

Usually, @options file and command line options are added

RosettaScripts protocol conventions

```
<ROSETTASCRIPTS>
        <SCOREFXNS>
        </SCOREFXNS>
        <RESIDUE_SELECTORS>
        </RESIDUE_SELECTORS>
        <FILTERS>
        </FILTERS>
        <TASKOPERATIONS>
        </TASKOPERATIONS>
        <MOVERS>
        </MOVERS>
        <APPLY_TO_POSE>
        </APPLY_TO_POSE>
        <PROTOCOLS>
        </PROTOCOLS>
        <OUTPUT/>
</ROSETTASCRIPTS>
```

- XML “eXtensible Markup Language”
- Consists of large level tags and sub-tags
- Widely used for representing hierarchical data
- Everything not in brackets < > is a comment

Tip: Run rosetta_scripts without options to get template

Breaking down a tag

```
<MOVERS>
   <PackRotamersMover name="repack1" scorefxn="score12_002" />
</MOVERS>
```

- Name of mover used
- Name assigned to specific version (can be referenced elsewhere in XML)
- Custom settings

Most tags have required settings or default values, always check documentation!

Movers – core of your protocol

```
<MOVERS>
    <PackRotamersMover name="repack1" scorefxn="score12_002" taskoperations="ifcl,rtrp" />
    <PackRotamersMover name="repack2" scorefxn="score12_005" taskoperations="ifcl,rtrp" />
    <PackRotamersMover name="repack3" scorefxn="score12_055" taskoperations="ifcl,rtrp" />
    <InterfaceAnalyzerMover name="iface" scorefxn="score12" fixedchains="A,B" />
</MOVERS>
```

- Movers are the basic building blocks of a RosettaScripts protocol
- Most modify the pose
  - Some compute metrics instead
- A single mover can be used more than once

Residue Selectors

```
<RESIDUE_SELECTORs>
   <Chain name="chA" chains="A"/>
   <Index name="res1to10" resnum="1-10"/>
</RESIDUE_SELECTORS>

<MOVERS>
   <PackRotamersMover name="repack1" taskoperations="rtrp" />
</MOVERS>
```

- Selects a subset of the system for Rosetta to operate on

- There are overlaps with other XML parts (example: a mover may define residues in its own way)

Score Functions

```
<SCOREFXNS>
    <ScoreFunction name="ligand_soft_rep" weights="ligand_soft_rep" />
    <ScoreFunction name="hard_rep" weights="ligand">
           <Reweight scoretype="fa_intra_rep" weight="0.004"/>
           <Reweight scoretype="fa_elec" weight="0.42"/>
</ScoreFunction>
</SCOREFXNS>
```

- Different parts of a protocol can use different score functions
- Standard score functions can be modified

Simple Metrics

```
<SIMPLE_METRICS>
    <RMSDMetric name="RMSD" residue_selector="align" residue_selector_ref="align_native"
robust="true" super="1" rmsd_type="rmsd_all" use_native="1"/>
</SIMPLE_METRICS>
```

- Simple Metrics can be used to score the pose during or after your run and retrieve data from your poses

Filters

```
<FILTERS>
     <ScoreType name="score_type_filter" scorefxn="score12" score_type="total_score"
Threshold="-500" />
     <AverageDegree name="avg_deg" threshold="8" distance_threshold="10"
     task_operations="rtiv" />
</FILTERS>
```

- Can pass/fail an output structure
  - Stop a run earlier if the output will be bad.

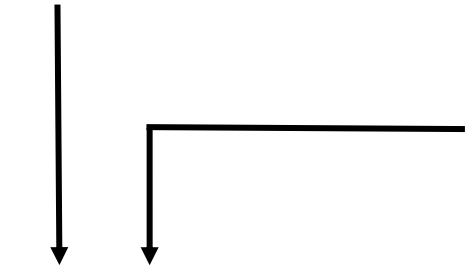
- Also can be used to compute protein metrics

Task Operations

```
<TASKOPERATIONS>
     <ReadResfile name="rrf" filename="resfile" />
     <RestrictToRepacking name="rtrp" />
     <RestrictResidueToRepacking name="restrict_Y100" resnum="100" />
</TASKOPERATIONS>

<MOVER>
     <PackRotamersMover name="repack1" taskoperations="rtrp" />
<MOVERS/>
```

PackRotamersMover
(repacks by default with every possible side chain, e.g. it does not care about the amino acid identity)



PackerTask

TaskOperations
Restrict the Packer to do what you want it to do
(select residues, define design tasks, etc.)

Protocols

```
<PROTOCOLS>
   <Add mover="Repack1"/>
   <Add mover="Repack2" filter="avg_deg" />
   <Add mover="iface"/>
</PROTOCOLS>
```

- Movers are executed in the order specified in PROTOCOLS
- Movers can be combined with filters
- Movers can be used more than once in a protocol

Output

```
<OUTPUT scorefxn="ref2015" />
```

- Specifies the score function used for the final output model and in the scorefile

- If you use multiple score functions in a protocol or use a non-default score function – make sure to flag this!

How to read a xml file – an easy case

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
    </SCOREFXNS>
    <RESIDUE_SELECTORS>
      <Index name="align" resnums="1-152"/>
      <Index name="align_native" resnums="1-152"/>
    </RESIDUE_SELECTORS>
    <SIMPLE_METRICS>
      <RMSDMetric name="RMSD" residue_selector="align" residue_selector_ref="align_native" robust="true" super="1"
rmsd_type="rmsd_all" use_native="1"/>
    </SIMPLE_METRICS>
    <TASKOPERATIONS>
    </TASKOPERATIONS>
    <FILTERS>
    </FILTERS>
    <MOVERS>
      <RunSimpleMetrics name="run_metrics1" metrics="RMSD" prefix="m1_" />
    </MOVERS>
    <APPLY_TO_POSE>
    </APPLY_TO_POSE>
    <PROTOCOLS>
      <Add mover="run_metrics1"/>
    </PROTOCOLS>
    <OUTPUT />
</ROSETTASCRIPTS>
```

How RosettaScripts parses protocols

- At initialization
  - Movers, filters, scoring functions, etc. are initialized
- For each input job
  - Movers and filters are executed in the order specified in PROTOCOLS
- Movers are **not aware** of one another
- Jobs are **not aware** of one another
- Only YOU are aware

Useful Features

- Rewrite old Rosetta XML scripts
  - tools/xsd_xrw/rewrite_rosetta_scripts.py

- Validate your XML scripts
  - https://www.rosettacommons.org/docs/latest/application_documentation/rosetta_scripts/validate_rosetta_script
  - Automatically runs when RosettaScripts starts

Variable substitution

/rosetta/main/source/bin/rosetta_scripts.default.linuxgccrelease
-parser:script_vars resfile=A105T.resfile -parser:protocol design.xml -s model.pdb

```
<ROSETTASCRIPTS>
    <SCOREFXNS>
      <ScoreFunction name="ref2015" weights="ref2015.wts" >
      </ScoreFunction>
    </SCOREFXNS>
    <TASKOPERATIONS>
      <InitializeFromCommandline name="ifcl" />
      <ReadResfile name="rrf" filename="%%resfile%%"/>
    </TASKOPERATIONS>
    <MOVERS>
      <PackRotamersMover name="design" scorefxn="ref2015" task_operations="ifcl,rrf" />
    </MOVERS>
    <FILTERS>
    </FILTERS>
    <APPLY_TO_POSE>
    </APPLY_TO_POSE>
    <PROTOCOLS>
      <Add mover="design" />
    </PROTOCOLS>
    <OUTPUT scorefxn="ref2015" />
</ROSETTASCRIPTS>
```

Documentation

RosettaScripts documentation

https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/RosettaScripts

Possible Movers

https://www.rosettacommons.org/docs/latest/scripting_documentation/RosettaScripts/Movers/Movers-RosettaScripts

Original reference

Fleishman, Sarel J., et al. "RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite." PloS one 6.6 (2011): e20161.