Rosetta Basics:
IO and Scripting

Joseph DeCorte
Graduate Student | Meiler Lab
joseph.a.decorte@vanderbilt.edu

Goals for this Talk

1. General Rosetta Concepts:
   - How do I run basic Rosetta applications?
   - Input/Output: file types, options, etc.

2. Learn where things are in Rosetta

3. Introduce RosettaScripts for custom protocol design

Please ask questions as they come up!

This talk is located in:
~/rosetta_workshop/tutorials/short_talks/navigating_rosetta_IO.pptx

Example files are located in:
Files for this talk are in ~/rosetta_workshop/tutorials/short_talks/RosettaIO/

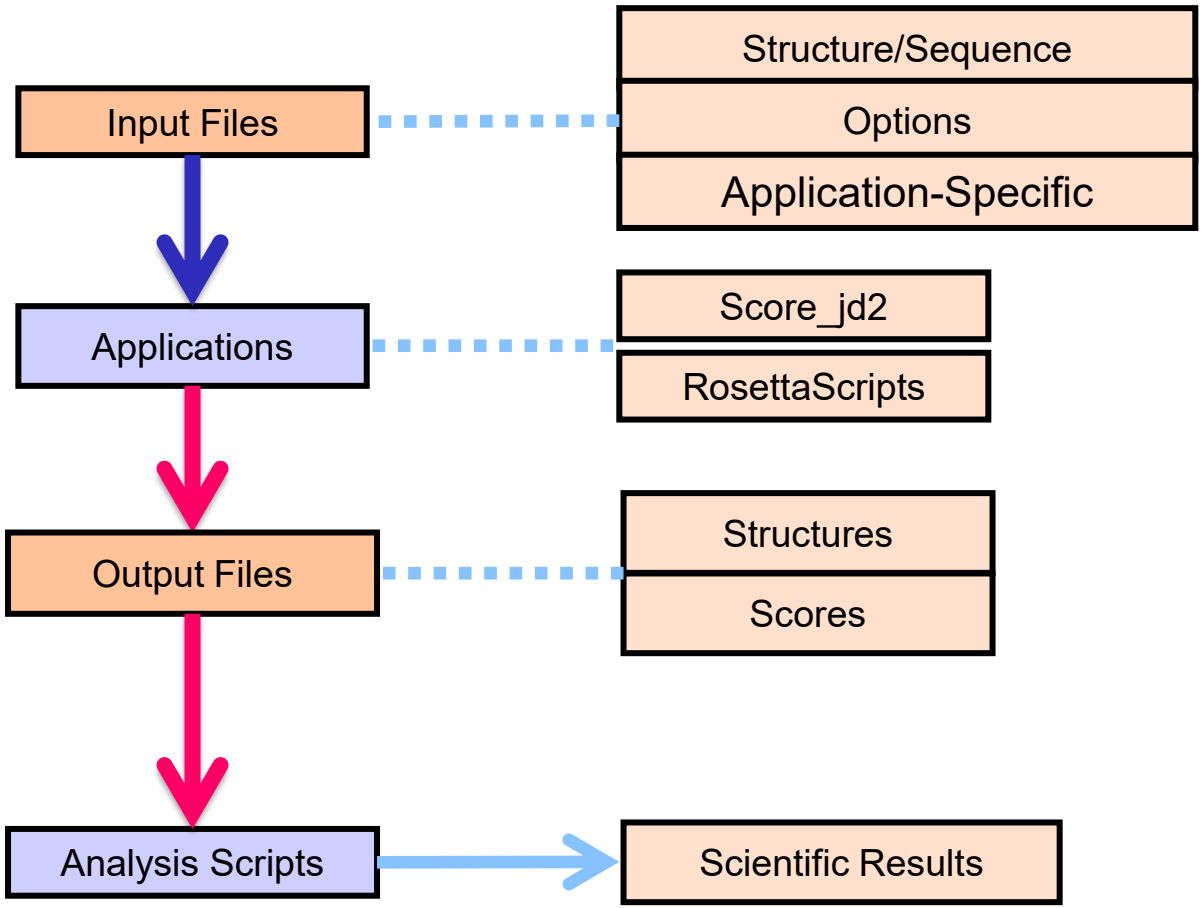
Notice these files are located outside of Rosetta--you do NOT want to store your input/output files in the Rosetta directories

Tip: Open each file in your favorite text editor (gedit, vim, emacs, etc.) as we introduce them through the talk

How do I get Rosetta?

- https://www.rosettacommons.org/software/license-and-download
- Weekly Releases: (e.g. “2020.07”)
  - Latest version of the code, released roughly every week
  - Every revision passes scientific performance tests
- Numbered Releases (e.g. “3.12”)
  - A weekly release that’s relabeled, released roughly every 6 months
- All tutorials use version 3.12
- Links to documentation, forum and demos:
  - https://www.rosettacommons.org/docs/latest/Home
  - https://www.rosettacommons.org/demos/latest/Home

General Workflow



How do I run a Rosetta command?

Every command has the same basic layout:

Path to the Rosetta application

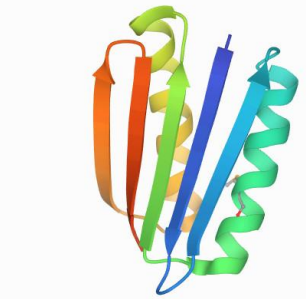
Arguments/flags/options

```
<path_to_rosetta>/main/source/bin/<app_name>.default.linuxgccrelease  –arg1 –arg2
```

Arguments consist of multiple things:

1. Point to input files
2. Point to where you want output files to go
3. Other arguments are protocol-dependent

A simple Rosetta command:



- cd ~/rosetta_workshop/short_talks/RosettaIO/
- Scoring 1qys (https://www.rcsb.org/structure/1QYS)
- Crystal structure of Top7: A computationally designed protein with a novel fold

<path_to_rosetta>/main/source/bin/score_jd2.default.linuxgccrelease –in:file:s 1qys.pdb -out:pdb > 1qys_score.log

- Inputs:
  - Running score_jd2 application, which simply scores in the input protein
  - -in:file:s 1qys.pdb : tells Rosetta we're inputting the 1qys.pdb PDB file
  - -out:pdb argument tells Rosetta that we want to save the output PDB file of the scored protein
  - > 1qys_score.log : print terminal output to file called 1qys_score.log
- Outputs:
  - 1qys_0001.pdb : output PDB
  - score.sc : default name for output scorefile
  - 1qys_score.log: tracer of run AKA what is output to the terminal screen when running command

Kuhlman B, Dantas G, Ireton GC, Varani G, Stoddard BL, Baker D. Design of a novel globular protein fold with atomic-level accuracy. *Science*. 2003;302(5649):1364-1368. doi:10.1126/science.1089427

Reading structures into Rosetta

PDB files

- International standard
- Readable by PyMol, MOE, Chimera, etc
- One line per atom
- Useful for small number of structures

www.wwpdb.org/documentation/file-forma

Common command line arguments

- Common input options:
  - -in:file:s example.pdb ## input a PDB structure file
  - -parser:protocol example.xml ## RosettaScripts XML file
  - -in:file:fasta example.fasta ## input a FASTA sequence file
  - -in:file:silent example.silent ## input a Rosetta silent file
  - -nstruct 42 ## produce 42 outputs
- Common output options:
  - -out:file:silent example_out.silent ## output structures to silent file
  - -out:file:scorefile example_out.sc ## output scorefile for run

Examples of output: the scorefile

score.sc

- One output per line—name of output is in the very last column
- Each column defines a specific score term for the respective output structure
- Second column is the "total_score"
- The following columns are individual scoreterms (described in detail in later talk)
- Excerpt of example scorefile here, but recommend you look at your own score.sc output file

```
SEQUENCE:
SCORE:     score     fa_atr     fa_rep     fa_sol    fa_intra_rep    fa_elec    ...     omega     fa_dun    p_aa_pp        ref                    description
SCORE:  -1217.209  -2778.696    266.309   1545.149           5.900   -301.320    ...    63.032    684.989   -109.110    -32.534      3gbm_HA_3gbn_Ab_full_0011
SCORE:  -1217.028  -2792.422    263.906   1549.738           5.867   -295.799    ...    66.036    682.694   -108.402    -32.534      3gbm_HA_3gbn_Ab_full_0012
SCORE:  -1204.280  -2760.354    259.175   1534.072           5.913   -293.050    ...    65.391    674.840   -108.393    -32.534      3gbm_HA_3gbn_Ab_full_0013
SCORE:  -1207.127  -2768.191    260.443   1541.857           5.881   -301.847    ...    67.951    686.381   -110.919    -32.534      3gbm_HA_3gbn_Ab_full_0014
SCORE:  -1208.390  -2769.872    262.398   1539.668           5.879   -297.571    ...    64.073    681.731   -109.633    -32.534      3gbm_HA_3gbn_Ab_full_0015
```

Examples of output: the output PDB

1qys_0001.pdb

- One atom/line just like normal PDBs

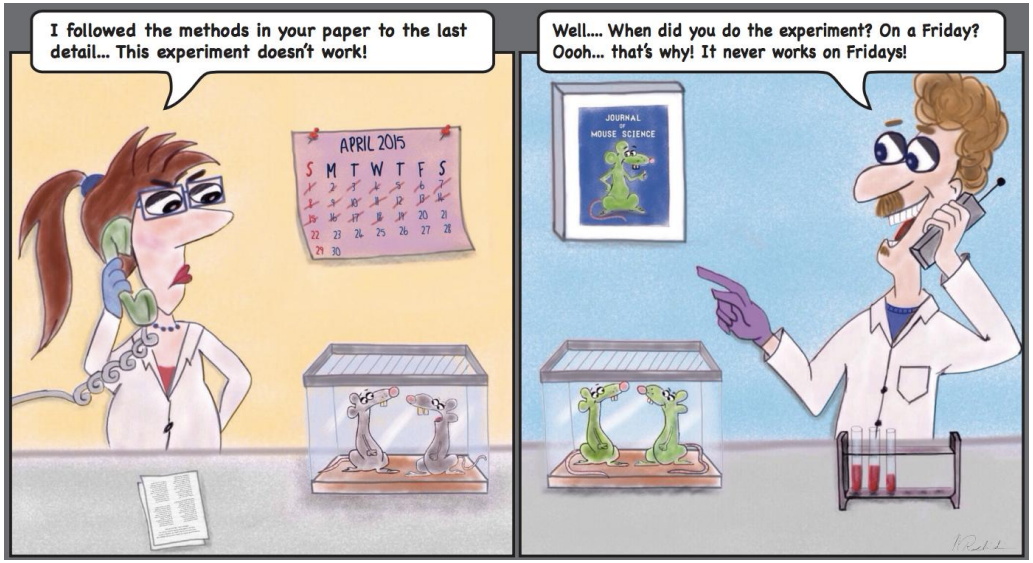
- Scroll to the bottom and there is per residue score information

```
ATOM   3378  HB  THR L 227     -36.166  22.580  28.848  1.00  0.00           H
ATOM   3379  HG1 THR L 227     -34.994  19.987  29.136  1.00  0.00           H
ATOM   3380 1HG2 THR L 227     -34.138  22.579  30.246  1.00  0.00           H
ATOM   3381 2HG2 THR L 227     -35.593  22.831  31.238  1.00  0.00           H
ATOM   3382 3HG2 THR L 227     -34.799  21.240  31.213  1.00  0.00           H
TER
# All scores below are weighted scores, not raw scores.
#BEGIN_POSE_ENERGIES_TABLE 3gbn_Ab_0005.pdb
label fa_atr fa_rep fa_sol fa_intra_rep fa_elec pro_close hbond_sr_bb hbond_lr_bb hbond_bb_sc hbond_sc dslf_fa13 rama omega fa_dun p_aa_pp yhh_planarity ref total
weights 1 0.55 0.9375 0.005 0.875 1.25 1.17 1.17 1.17 1.1 1.25 0.25 0.625 0.7 0.4 0.625 1 NA
pose -994.338 137.719 561.027 2.15688 -112.612 19.7634 -12.2069 -79.5391 -24.1449 -23.4441 -1.15166 -7.47192 71.8572 276.633 -29.8673 0.09431 13.9828 -201.541
GLU:NtermProteinFull_1 -1.58225 0.53996 1.45283 0.00353 0.06909 0 0 0 0 0 0 0 0.01109 6.53174 0 0 -1.96094 5.06505
VAL_2 -3.34255 0.45648 1.46378 0.01322 -0.08167 0 0 0 0 0 0 -0.16095 0.87346 0.30715 0.39992 0 0.97964 0.90848
GLN_3 -2.79445 0.10936 1.74929 0.00451 -0.52743 0 0 0 -0.35772 0 0 -0.09682 0.35321 2.59775 0.02034 0 -1.51717 -0.45911
LEU_4 -5.13483 0.73792 1.6574 0.00685 -0.16379 0 0 0 0 0 0 0.06265 0.2281 2.29891 -0.1217 0 0.76113 0.33264
VAL_5 -2.72905 0.12167 1.72074 0.00789 -0.45069 0 0 0 0 0 0 -0.27382 0.01969 0.02557 -0.49649 0 0.97964 -1.07485
```

Examples of output: Tracer output (log files)

1qys_score.log

- Shows exactly the command line you are running at the beginning
- Which databases are being used, calling protocols, warnings, errors, etc.
- Very useful for debugging to figure out where problems are coming from
- Makes your protocol reproducible!!



Examples of output: Tracer output (log files)

1qys_score.log

```
core.init: Rosetta version exported  from http://www.rosettacommons.org
core.init: command: /dors/meilerlab/apps/rosetta/rosetta_2016.08.58479/main/source/bin/rosetta_scripts.default.linuxgccrelease @docking.options -parser:protocol docking
core.init: 'RNG device' seed mode, using '/dev/urandom', seed=1059677151 seed_offset=0 real_seed=1059677151
core.init.random: RandomGenerator:init: Normal mode, seed=1059677151 RG_type=mt19937
core.init: Resolved executable path: /dors/meilerlab/apps/rosetta/rosetta_2016.08.58479/main/source/build/src/release/linux/2.6/64/x86/gcc/5.2/default/rosetta_scripts.d
core.init: Looking for database based on location of executable: /dors/meilerlab/apps/rosetta/rosetta_2016.08.58479/main/database/
protocols.jd2.PDBJobInputter: Instantiate PDBJobInputter
protocols.jd2.PDBJobInputter: PDBJobInputter::fill_jobs
protocols.jd2.PDBJobInputter: pushed 3gbm_HA_3gbn_Ab.pdb nstruct indices 1 - 50
protocols.evaluation.ChiWellRmsdEvaluatorCreator: Evaluation Creator active ...
protocols.jd2.JobDistributor: Parser is present.  Input mover will be overwritten with whatever the parser creates.
protocols.jd2.PDBJobInputter: PDBJobInputter::pose_from_job
protocols.jd2.PDBJobInputter: filling pose from PDB 3gbm_HA_3gbn_Ab.pdb
core.chemical.ResidueTypeSet: Finished initializing fa_standard residue type set.  Created 384 residue types
core.chemical.ResidueTypeSet: Total time to initialize 0.43 seconds.
core.conformation.Conformation: Found disulfide between residues 7 461
...
protocols.rosetta_scripts.ParsedProtocol.REPORT: ============End report for ==================
protocols.rosetta_scripts.ParsedProtocol.REPORT: ============Begin report for ==================
protocols.rosetta_scripts.ParsedProtocol.REPORT: ============End report for ==================
protocols.jd2.JobDistributor: 3gbm_HA_3gbn_Ab_full_0050 reported success in 381 seconds
protocols.jd2.JobDistributor: no more batches to process...
protocols.jd2.JobDistributor: 50 jobs considered, 50 jobs attempted in 16297 seconds
~
~
~
```

Options to control tracer output *these files can get very long!*:

- Silence certain tracers:

  -mute core.chemical.ResidueTypeSet

- Change verbosity level (Error/Warning/Info/Debug/Trace)

  -out:levels all:Warning core.init:Info

Other files: application-specific

- Span File: Defines which residues are in the membrane
- Loops File: Identifies the loop residues for loop closure
- Params File: Custom parameters for small molecules or non-canonical amino acids
- Constraint File: Experimentally derived restraints
- Fragment File: Short protein segments used for comparative modeling and de novo folding
- Res Files: Indicates which residue positions should be designed

Protocols can get complicated…

Toward high-resolution prediction and design of transmembrane helical protein structures

P. Barth, J. Schonbrun*, and D. Baker†

Department of Biochemistry and Howard Hughes Medical Institute, University of Washington, Seattle, WA 98195

```
$ROSETTA/main/source/bin/AbinitioRelax.linuxgccrelease -database
../../rosetta_database -in:file:fasta ./input_files/1elwA.fasta -
in:file:native ./input_files/1elw.pdb -in:file:frag3
./input_files/aa1elwA03_05.200_v1_3 -in:file:frag9
./input_files/aa1elwA09_05.200_v1_3 -abinitio:relax -relax:fast -
abinitio::increase_cycles 10 -abinitio::rg_reweight 0.5 -
abinitio::rsd_wt_helix 0.5 -abinitio::rsd_wt_loop 0.5 -use_filters
true -psipred_ss2 ./input_files/1elwA.psipred_ss2 -kill_hairpins -
out:file:silent 1elwA_silent.out
-nstruct 10
```

OR

```
$ROSETTA/main/source/bin/AbinitioRelax.linuxgccrelease @options.txt
```

Use an options file for your runs

Why?

- Easier to read/organize
- Reproducibility!

```
$ROSETTA/main/source/bin/AbinitioRelax.linuxgccrelease @options.txt
```

```
-in:file
        -fasta ./input_files/1e1wA.fasta
        -native ./input_files/1e1w.pdb
        -frag3 ./input_files/aa1elwA03_05.200_v1_3
        -frag9 ./input_files/aa1elwA09_05.200_v1_3
-psipred_ss2 ./input_files/1elwA.psipred_ss2
-abinitio:relax
-relax:fast
-abinitio::increase_cycles 10
-abinitio::rg_reweight 0.5
…
-out:file:silent ./output_files/1elwA_silent.out
-nstruct 10
```

Any Questions?

Rosetta Basics:
IO and **Scripting**

Joseph DeCorte
Graduate Student | Meiler Lab
joseph.a.decorte@vanderbilt.edu

Rosetta applications do not cover all protocols

```
ls /rosetta-3.11/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 must be tested and show evidence that they fulfill their task (benchmarking)

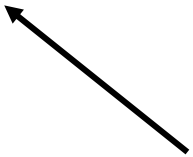
- Technical:
  - Not all XMLs/options have been tested in combination
  - Some XMLs require specific options to work
  - Some tinkering expected…

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

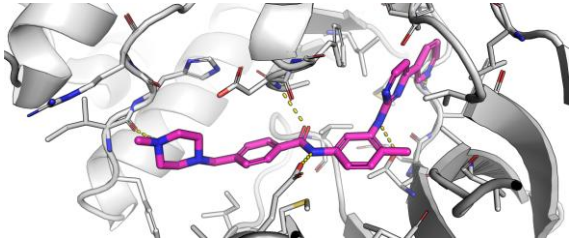
```
<ROSETTASCRIPTS>
        <SCOREFXNS>
        </SCOREFXNS>
        <RESIDUE_SELECTORS>
        </RESIDUE_SELECTORS>
        <FILTERS>
        </FILTERS>
        <TASKOPERATIONS>
        </TASKOPERATIONS>
        <MOVERS>
        </MOVERS>
        <APPLY_TO_POSE>
        </APPLY_TO_POSE>
        <PROTOCOLS>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

Learning by example: Ligand Docking

```
<ROSETTASCRIPTS>

    <SCOREFXNS>
        <ScoreFunction name="ligand_soft_rep" weights="ligand_soft_rep">
        </ScoreFunction>
        <ScoreFunction name="hard_rep" weights="ligand">
        </ScoreFunction>
    </SCOREFXNS>

    <LIGAND_AREAS>
        <LigandArea name="inhibitor_dock_sc" chain="X" cutoff="6.0" add_nbr_radius="true" all_atom_mode="false"/>
        <LigandArea name="inhibitor_final_sc" chain="X" cutoff="6.0" add_nbr_radius="true" all_atom_mode="true"/>
        <LigandArea name="inhibitor_final_bb" chain="X" cutoff="7.0" add_nbr_radius="false" all_atom_mode="true" Calpha_restraints="0.3"/>
    </LIGAND_AREAS>

    <INTERFACE_BUILDERS>
        <InterfaceBuilder name="side_chain_for_docking" ligand_areas="inhibitor_dock_sc"/>
        <InterfaceBuilder name="side_chain_for_final" ligand_areas="inhibitor_final_sc"/>
        <InterfaceBuilder name="backbone" ligand_areas="inhibitor_final_bb" extension_window="3"/>
    </INTERFACE_BUILDERS>

    <MOVEMAP_BUILDERS>
        <MoveMapBuilder name="docking" sc_interface="side_chain_for_docking" minimize_water="false"/>
        <MoveMapBuilder name="final" sc_interface="side_chain_for_final" bb_interface="backbone" minimize_water="false"/>
    </MOVEMAP_BUILDERS>

    <SCORINGGRIDS ligand_chain="X" width="20">
        <ClassicGrid grid_name="classic" weight="1.0"/>
    </SCORINGGRIDS>

    <MOVERS>
        <Transform name="transform" chain="X" box_size="7.0" move_distance="0.2" angle="20" cycles="500" repeats="1" temperature="5"/>
        <HighResDocker name="high_res_docker" cycles="6" repack_every_Nth="3" scorefxn="ligand_soft_rep" movemap_builder="docking"/>
        <FinalMinimizer name="final" scorefxn="hard_rep" movemap_builder="final"/>
        <InterfaceScoreCalculator name="add_scores" chains="X" scorefxn="hard_rep"/>
    </MOVERS>

    <PROTOCOLS>
        <Add mover_name="transform"/>
        <Add mover_name="high_res_docker"/>
        <Add mover_name="final"/>
        <Add mover_name="add_scores"/>
    </PROTOCOLS>
</ROSETTASCRIPTS>
```



PDB: 2HYY. Image made with PyMOL.

Breaking down a tag

```
<MOVERS>
    <Transform name="transform" chain="X" box_size="7.0" move_distance="0.2"
 angle="20" cycles="500" repeats="1" temperature="5"/>
    ... // Some Movers omitted for clarity
 </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!

Protocols

```
<PROTOCOLS>
        <Add mover_name="transform"/>
        <Add mover_name="high_res_docker"/>
        <Add mover_name="final"/>
        <Add mover_name="add_scores"/>
</PROTOCOLS>
```

- PROTOCOLS define your order of operations via **movers**
- Movers can be combined with **filters**
- Movers can be used more than once in a protocol

Movers – the core of your protocol

```
<MOVERS>
    <Transform name="transform" chain="X" box_size="7.0" move_distance="0.2"
angle="20" cycles="500" repeats="1" temperature="5"/>
    <HighResDocker name="high_res_docker" cycles="6" repack_every_Nth="3"
scorefxn="ligand_soft_rep" movemap_builder="docking"/>
    <FinalMinimizer name="final" scorefxn="hard_rep" movemap_builder="final"/>
    <InterfaceScoreCalculator name="add_scores" chains="X" scorefxn="hard_rep"/>
</MOVERS>
```

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

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 (see *Reweight*)

Term	Description	Weight	Units	Ref.
fa_atr	Attractive energy between two atoms on different residues separated by distance, *d*	1.0	kcal/mol	[5,6]
fa_rep	Repulsive energy between two atoms on different residues separated by distance, *d*	0.55	kcal/mol	[5,6]
fa_intra_rep	Repulsive energy between two atoms on the same residue, separated by distance, *d*	0.005	kcal/mol	[5,6]
fa_sol	Gaussian exclusion implicit solvation energy between protein atoms in different residues	1.0	kcal/mol	[36]
lk_ball_wtd	Orientation-dependent solvation of polar atoms assuming ideal water geometry	1.0	kcal/mol	[50,71]
fa_intra_sol	Gaussian exclusion implicit solvation energy between protein atoms in the same residue	1.0	kcal/mol	[36]
fa_elec	Energy of interaction between two non-bonded charged atoms separated by distance, *d*	1.0	kcal/mol	[50]

The Rosetta all-atom energy function for macromolecular modeling and design. JCTC 2018.

Putting it all together

```
<ROSETTASCRIPTS>

    <SCOREFXNS>
        <ScoreFunction name="ligand_soft_rep" weights="ligand_soft_rep">
        </ScoreFunction>
        <ScoreFunction name="hard_rep" weights="ligand">
        </ScoreFunction>
    </SCOREFXNS>

    <LIGAND_AREAS>
        <LigandArea name="inhibitor_dock_sc" chain="X" cutoff="6.0" add_nbr_radius="true" all_atom_mode="false"/>
        <LigandArea name="inhibitor_final_sc" chain="X" cutoff="6.0" add_nbr_radius="true" all_atom_mode="true"/>
        <LigandArea name="inhibitor_final_bb" chain="X" cutoff="7.0" add_nbr_radius="false" all_atom_mode="true" Calpha_restraints="0.3"/>
    </LIGAND_AREAS>

    <INTERFACE_BUILDERS>
        <InterfaceBuilder name="side_chain_for_docking" ligand_areas="inhibitor_dock_sc"/>
        <InterfaceBuilder name="side_chain_for_final" ligand_areas="inhibitor_final_sc"/>
        <InterfaceBuilder name="backbone" ligand_areas="inhibitor_final_bb" extension_window="3"/>
    </INTERFACE_BUILDERS>

    <MOVEMAP_BUILDERS>
        <MoveMapBuilder name="docking" sc_interface="side_chain_for_docking" minimize_water="false"/>
        <MoveMapBuilder name="final" sc_interface="side_chain_for_final" bb_interface="backbone" minimize_water="false"/>
    </MOVEMAP_BUILDERS>

    <SCORINGGRIDS ligand_chain="X" width="20">
        <ClassicGrid grid_name="classic" weight="1.0"/>
    </SCORINGGRIDS>

    <MOVERS>
        <Transform name="transform" chain="X" box_size="7.0" move_distance="0.2" angle="20" cycles="500" repeats="1" temperature="5"/>
        <HighResDocker name="high_res_docker" cycles="6" repack_every_Nth="3" scorefxn="ligand_soft_rep" movemap_builder="docking"/>
        <FinalMinimizer name="final" scorefxn="hard_rep" movemap_builder="final"/>
        <InterfaceScoreCalculator name="add_scores" chains="X" scorefxn="hard_rep"/>
    </MOVERS>

    <PROTOCOLS>
        <Add mover_name="transform"/>
        <Add mover_name="high_res_docker"/>
        <Add mover_name="final"/>
        <Add mover_name="add_scores"/>
    </PROTOCOLS>
</ROSETTASCRIPTS>
```

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>
```

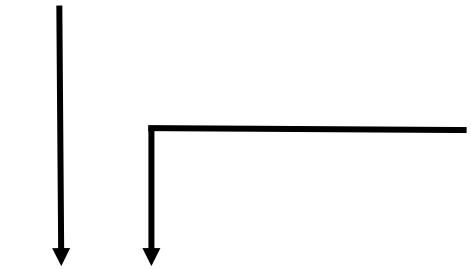
- Can pass/fail an output structure
  - Stop a run earlier if the output will be bad.

- Also can be used to compute protein metrics

Fine-Tuning with 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, i.e. it does not care about the amino acid identity)



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

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)

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

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>
```

Other 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

Documentation

RosettaScripts documentation

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

Descriptions of 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.