

This protocol capture will detail the how to use ROSETTADESIGNto predict mutations that enhance specificity. This accompanies the manuscript Willis *et al.* ***Nature Med. Letters*** (submitted). It assumes that you have a ROSETTA license from [www.rosettacommons.org](http://www.rosettacommons.org), and have downloaded and compiled the applications.

### Preparing the input files:

Using PG9/CAP45 complex, I have prepared a ROSETTA compatible file called PG9\_input.pdb. This has special identifiers for the glycans that ROSETTA's database can understand. To create your own glycan input, an excellent protocol capture is provided in an accompanying manuscript by Doug Renfrew (Renfrew et al., 2012).

The design protocol used runs through the following steps:

- Favor native residue - gives bonuses to sequences which match PG9*wt*
- Design/minimize/dock iteratively
- Constrain movements so glycans retain input position
- Relax the energy of the structure
- Score binding energy and structure energy

For this redesign we need several input files. The XML script, options file, residue file, and constraint file. The most complex of which, the XML file, informs ROSETTAof our protocol. Use the following .xml file which is found under

/input\_files/threading\_design.xml

.  
*XML-File*

```
<dock_design>
  <SCOREFXNS>
    Redefine scoring function to take in constraints
    <scorewts weights=talaris2013>
      <Reweight scoretype=atom_pair_constraint weight=0.5/>
    </scorewts>
    <scoredockwts weights=talaris2013 patch=docking>
      <Reweight scoretype=atom_pair_constraint weight=0.5/>
    </scoredockwts>
  </SCOREFXNS>
  <FILTERS>
    DDG filter for design – will design until this
    is satisfied
    <Ddg name=ddg chain_num=2,3 repack=1
      scorefxn=talaris2013 threshold=-20/>
    When docking or minimizing, won't violate atom-pairs defined
```

```

    in glycan_constraints
    <ScoreType name=atom_pair_constraint scorefxn=scoredockwts
      score_type=atom_pair_constraint threshold=100/>
  </FILTERS>
<TASKOPERATIONS>
  <InitializeFromCommandline name=ifcl/>
  <ReadResfile name=rrd filename="input_files/normal_design.resfile"/>
</TASKOPERATIONS>
<MOVERS>
  Gives bonuses for input residues
  <FavorSequenceProfile name=fsp scaling="prob"
    weight=1.5 use_current=1
    matrix="IDENTITY" scorefxns=scorewts/>
  Penalizes movements to far away from
  glycan
  <ConstraintSetMover name=pair_on
    cst_file="input_files/glycan_constraints.cst"/>
  Design step that takes in residue file
  <PackRotamersMover name=pr_design scorefxn=scorewts
    task_operations=rrd,ifcl/>
  Turns of penalization
  <ConstraintSetMover name=pair_off cst_file=none/>
  Docks protein around interface
  <Docking name=dock score_high=scoredockwts fullatom=1
    local_refine=1 jumps=1 task_operations=ifcl/>
  Docks until MonteCarlo criterion is satisfied
  <GenericMonteCarlo name=gmc_dock mover_name=dock
    filter_name=atom_pair_constraint drift=0/>
  Minimize energy of protein
  <MinMover name=min scorefxn=scorewts chi=1 bb=1 jump=1/>
  Design protocol
  <ParsedProtocol name=design_pp>
    <Add mover=pair_on/>
    <Add mover=pr_design/>
    <Add mover=gmc_dock/>
    <Add mover=min/>
    <Add mover=pair_off/>
  </ParsedProtocol>
  Run design until binding energy threshold is satisfied
  <GenericMonteCarlo name=gmc_design
    mover_name=design_pp filter_name=ddg drift=False/>
  Relax protein
  <FastRelax name=fr scorefxn=scorewts
    task_operations=ifcl/>
  Get DDG

```

```

<ddG name=per_ddg chain_name="H,L"
  scorefxn=talaris2013 per_residue_ddg=1/>
</MOVERS>
<APPLY_TO_POSE>
</APPLY_TO_POSE>
<PROTOCOLS>
  Ordered list of steps for the protocol
  each defined in the mover or filter definitions
  <Add mover_name=fsp/>
  <Add mover_name=gmc_design/>
  <Add mover_name=pair_on/>
  <Add mover_name=fr/>
  <Add mover_name=pair_off/>
  <Add mover_name=per_ddg/>
  <Add filter_name=ddg/>
</PROTOCOLS>
</dock_design>

```

The behavior of these instructions is described fully in (Fleishman et al., 2011). They are divided up into a set of movers, filters and task of operations. All of the movers and filters along with their options are explained at the Rosetta Commons users guide (<https://www.rosettacommons.org/docs/latest/RosettaScripts.html>).

#### *Options-File*

The options file are passed to the application. Defines output and input options as well as other options which can't be defined in the XML file.

```

-s input_files/pg9_input.pdb #input PDB
-nstruct 200 #the number of output models to generate
-docking
  -sc_min #minimize side chains during docking
-parser:protocol input_files/threading_design.xml
-ex1 #Rotmer library 1
-ex2 #Rotmer library 2
-ex1aro #Rotamer aromatic 1
-out:path
  -pdb ./output_files/
  -score /dev/null/

```

Each option is explained with a # comment.

#### *Residue File*

The residue file tells the packer how to design the protein. The first line lets the packer use the side chains of the input PDB even if they are not in the rotamer libraries. The "NATAA" lines tells the packer to use input amino acid for everything not defined under start. In other words it will only design everything under start. The first column is the residue number, the second is the chain, and "ALLAA" tells the packer to use all

amino acid identities at this position. For complete documentation of the resfile, visit <https://www.rosettacommons.org/docs/latest/resfiles.html>

USE.INPUT.SC

NATAA

EX 1 EX 2

start

96 H ALLAA

97 H ALLAA

98 H ALLAA

99 H ALLAA

100 H ALLAA

101 H ALLAA

102 H ALLAA

103 H ALLAA

104 H ALLAA

105 H ALLAA

106 H ALLAA

107 H ALLAA

108 H ALLAA

109 H ALLAA

110 H ALLAA

111 H ALLAA

112 H ALLAA

113 H ALLAA

114 H ALLAA

115 H ALLAA

116 H ALLAA

117 H ALLAA

118 H ALLAA

119 H ALLAA

120 H ALLAA

121 H ALLAA

122 H ALLAA

123 H ALLAA

124 H ALLAA

125 H ALLAA

*Constraint File*

The constraint file ensures that the glycan's are involved in binding. The torsional angles of the glycan can cause major structural perturbations.

AtomPair NZ 57H O31 29G BOUNDED 0 2.57 0.2 0.5 tag

AtomPair O 54H O32 29G BOUNDED 0 3.71 0.2 0.5 tag

AtomPair O 55H OS1 29G BOUNDED 0 4.30 0.2 0.5 tag

AtomPair ND2 73H O71 29G BOUNDED 0 4.02 0.2 0.5 tag

```
AtomPair OD1 52L O81 33G BOUNDED 0.5 2.96 0.2 0.5 tag
AtomPair OG 34L O81 33G BOUNDED 0.5 2.28 0.2 0.5 tag
AtomPair NH2 30H NZ 41G BOUNDED 0 4.85 0.2 0.5 tag
```

The constrain file syntax is found in the documentation - (<https://www.rosettacommons.org/docs/latest/constraint-file.html>). Briefly, I define two atoms with the input crystal structure distances. If these are violated, then there is an energetic penalty.

## Running Rosetta

To run ROSETTA, I use an application called RosettaScripts (Fleishman et al., 2011). Since we have defined all the input files. Running the application only requires passing the options file.

```
my/path/to/rosetta/source/bin/rosetta_scripts.myoperatingsystem \
@input_files/threadint.txt
-database my/path/to/rosetta/database/
```

This protocol generates 200 models each taking approximately 1 hour to complete. It is best to run this protocol on a computational cluster with each node producing a separate model (-nstruct 1). All files are output into a directory output\_models/. There are 200 pre-generated models for analysis.

## Analyzing Models

There are three scripts in the /analysis folder that are used to analyze the mutations. Score\_vs\_rmsd\_full.py will give all the models energies as well as how much they deviated from the original structure. Get\_per\_ddg.py will give all of the binding energies decomposed by residues. Scores\_decomposed\_by\_resfile.py will decompose the energies of HCDR3 loop. They are each run using the following.

```
score_vs_rmsd_full.py -m -n ../input_files/pg9_input.pdb \
-o s_v_rmsd -r ../input_files/normal_design \
../output_files/*.pdb
```

```
get_per_ddg.py -m -o per_ddg \
../output_files/*.pdb
```

```
scores_decompose_by_resfile.py -m -o HCDR3 \
-r ../input_files/normal_design.res
```

These will yield a series of data files that can be uploaded to a database or in a spreadsheet viewer. The complex queries I used to check energies between *wt* and mutations are beyond the scope of a protocol capture. But you can contact jwillis0720@gmail.com if you need additional guidance.

# Bibliography

- Fleishman, S. J., Leaver-Fay, A., Corn, J. E., Strauch, E.-M., Khare, S. D., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., Meiler, J., and Baker, D. (2011). RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. *PloS one*, 6(6):e20161–.
- Renfrew, P. D., Choi, E. J., Bonneau, R., and Kuhlman, B. (2012). Incorporation of non-canonical amino acids into Rosetta and use in computational protein-peptide interface design. *PloS one*, 7(3):e32637.