

Protein Design

Benjamin K. Mueller

Postdoc, Meiler Lab

Rosetta Workshop

December 2017

Overview

Introduction to Protein Design

How it works:

The Packer

Resfiles

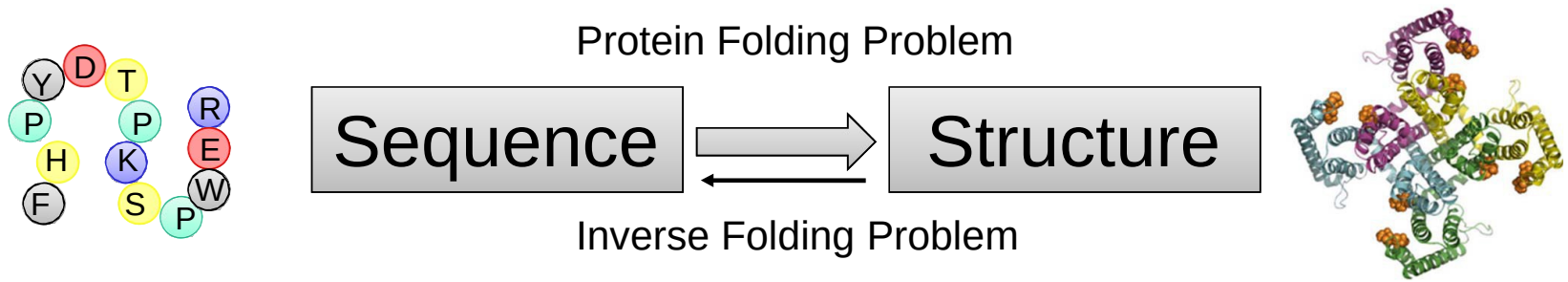
Overview of the Tutorial:

Input files

Protocol

Analysis

Protein Design is the Inverse Protein Folding Problem

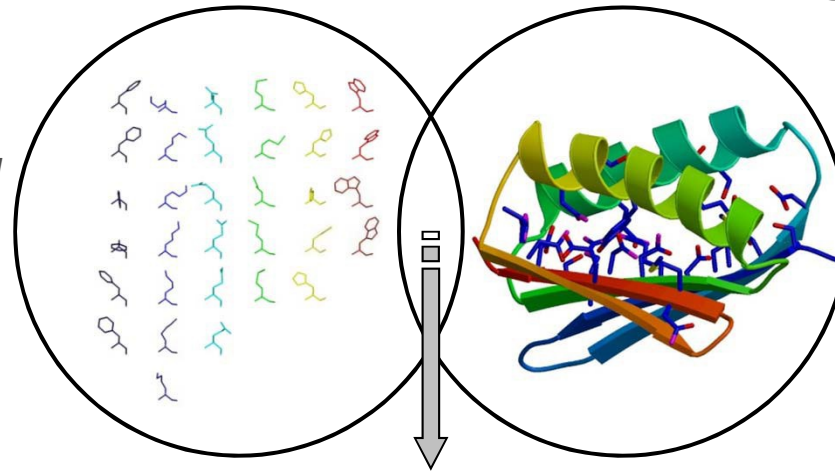


Given a protein fold – which primary sequence(s) can fold into it?

Protein Design Uses the Rosetta Energy Function and Local Rotamer Libraries

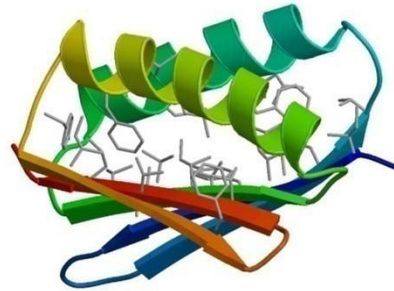
Local Rotamer Bias

Approximate interactions between sidechains using the distribution of sidechain conformations seen in known protein structures



Energy function

- VDW interactions
- solvation
- hydrogen bonding potential
- elec interactions
- rotamer probability



***Simulated Annealing
Monte Carlo optimization***

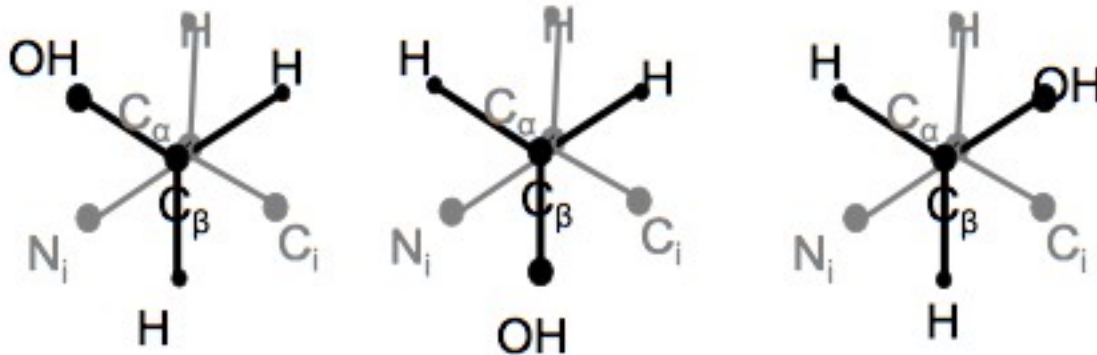
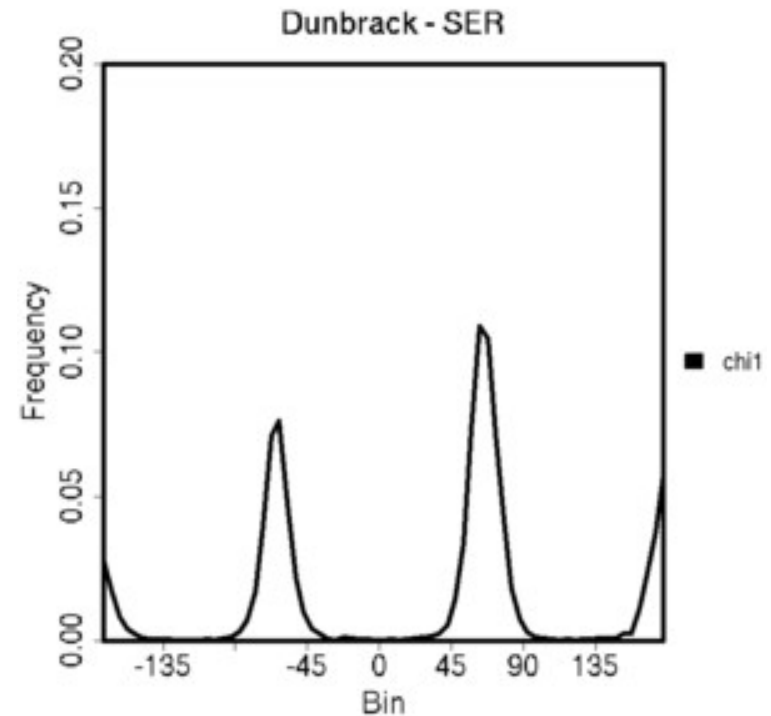
How Rosetta Design works...

Side chain rotamer optimization

Rotamer = **rotational isomer**

Sampling all chi angles is computationally expensive

Rotamers fall into discrete bins
– can simplify this with a rotamer library



The Packer

In general, the purpose is to optimize rotamers on a fixed backbone.

There are 3 general strategies for design:

- Native (repack)

- All Amino Acids (design)

- Specified Amino Acids (guided design)

The Packer – What it's actually doing

Steps of the packer:

1. Build rotamers & detect neighbors
2. Calculate rotamer energies

Loop steps 3-5:

3. Random rotamer substitution
4. Evaluate energy
5. Accept/Reject
 - Metropolis criterion
 - Simulated annealing
6. Return best energy

How to Control the Packer...

The Residue File Guides Design

ALLAA	allow all 20 amino
ALLAAwc	allow all 20 amino(default)
ALLAAxc	allow all amino acids except cysteine
POLAR	allow only canonical polar amino acids
APOLAR	allow only canonical non-polar amino acids
NOTAA	disallow only the specified amino acids
PIKAA	allow only the specified amino acids
NATAA	allow only the native amino acid (repack)
NATRO	preserve input rotamer
EMPTY	disallow all canonical amino acids
NC <ResName>	allow the specific non canonical residue

You can also combine commands (see tutorial).

```
25 A POLAR NOTAA K
```

When on separate lines (first specify a range, then specific residues):

```
5-20 A ALLAA  
15 A PIKAA Y
```

Residue 15 on chain A will only sample Y, whereas 5-14 and 16-20 will sample all

https://www.rosettacommons.org/docs/latest/rosetta_basics/file_types/resfiles

Basic Format of Residue Files

```
<Header> #instructions for all positions not specified in body
#The header can also use commands such as EX 1, EX 2,
#and USE_INPUT_SC to apply to all positions not specified below
START #keyword
<Body> #instructions for specific chains and identifiers
<PDBNUM> <CHAIN> <COMMANDS>
#Basic format for lines in body
* <CHAIN> <COMMANDS>
#used to specify a command for an entire chain
```

Residue File Example

Resfile that designs everything:

```
ALLAA  
START
```

Resfile that only repacks chain H:

```
NATRO  
START  
* H NATAA
```

Resfile that does nothing:

```
NATRO #keeps all input rotamers (and hence identity)  
START
```

What will the Packer do?

NATRO

START

* A NATAA

What will the Packer do?

NATRO

EX 1

START

30 A PIKAA FY

20-35 A ALLAAxc

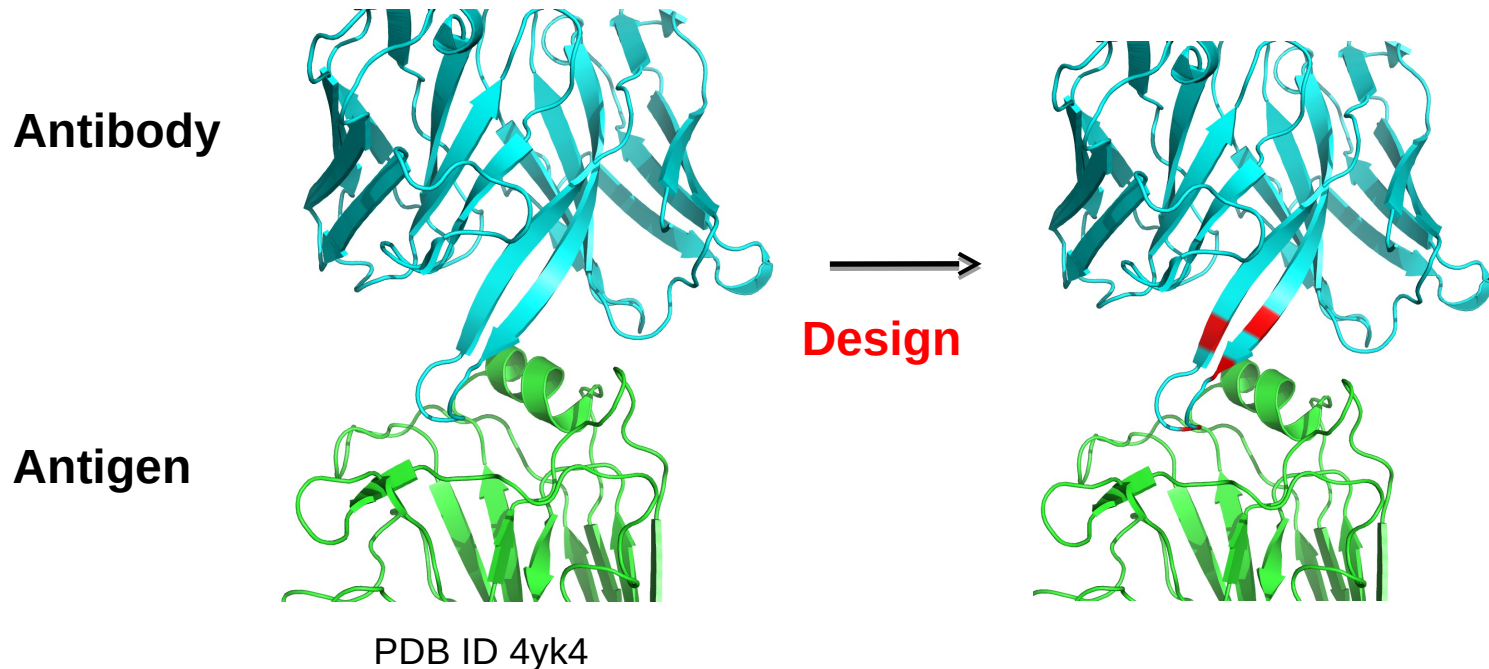
Tutorial Overview – Protein design in Rosetta

Antibody single-state design

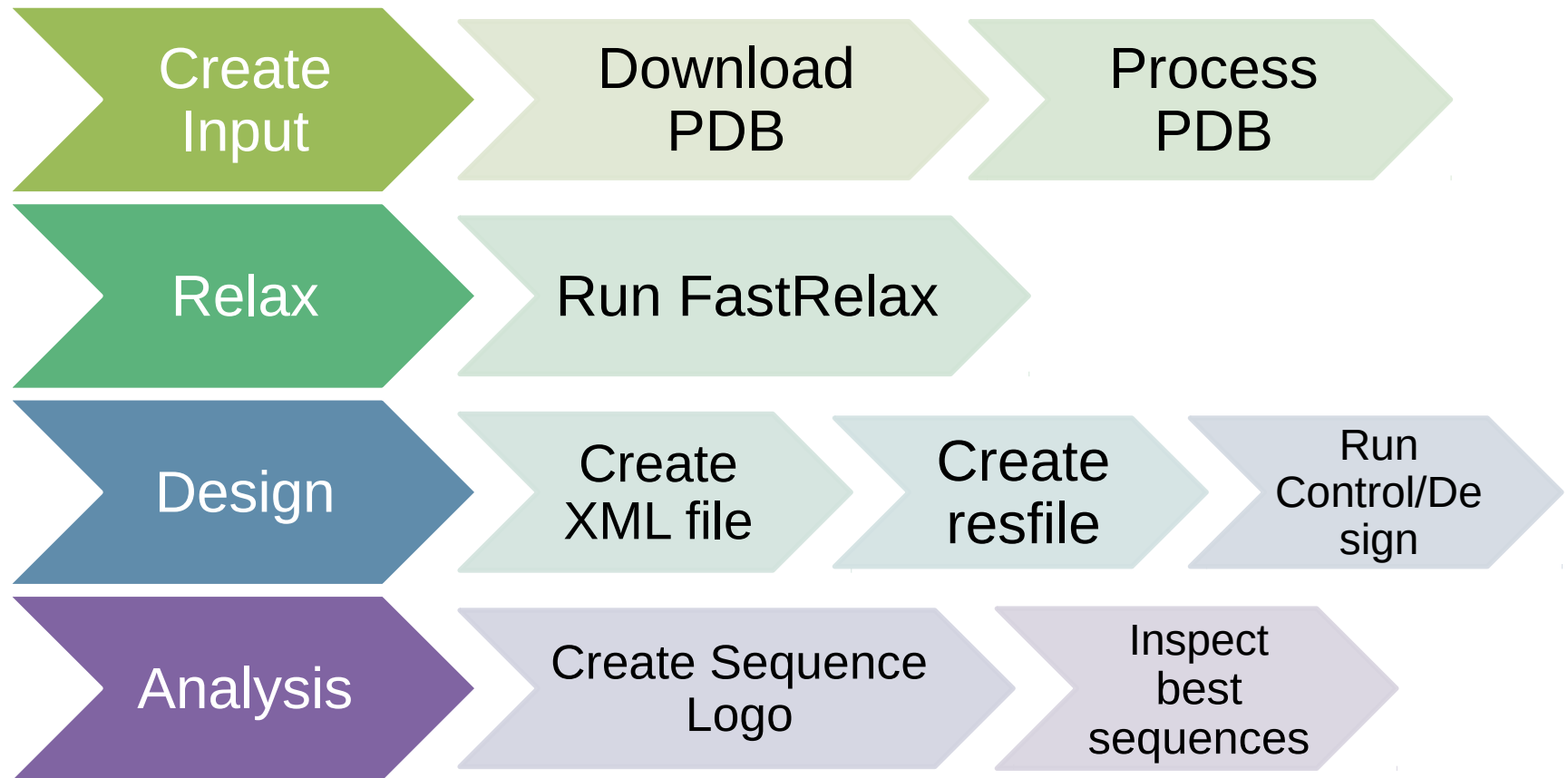
Antibody single-state design

Also known as redesign, computational affinity maturation

Goal: take an existing antibody-antigen complex and optimize the antibody sequence for tighter binding



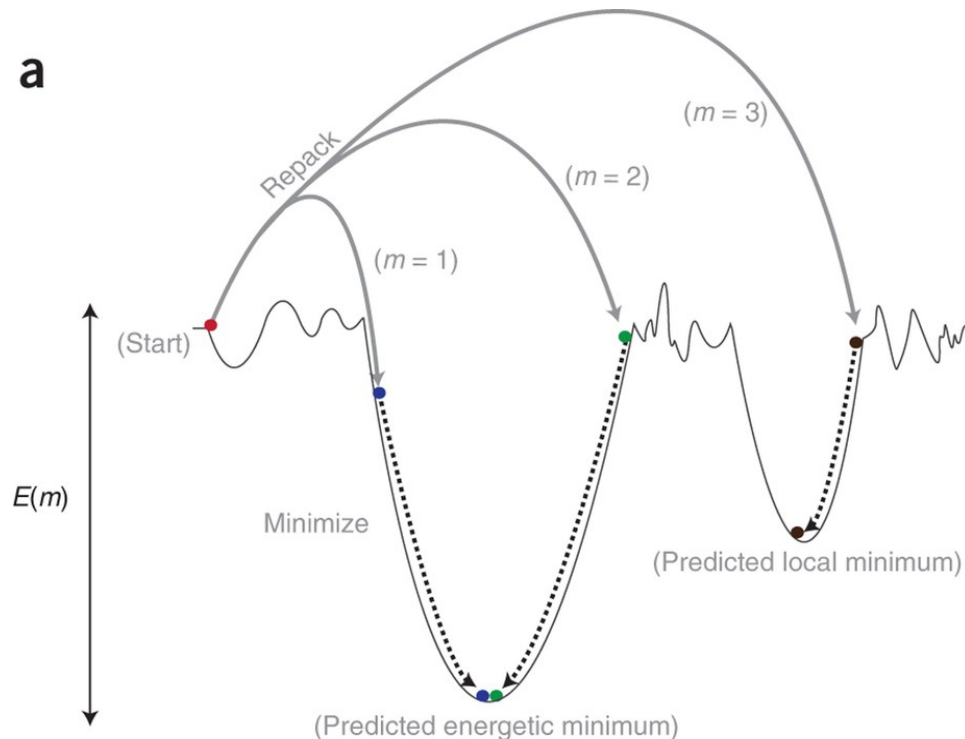
Single-state design protocol overview



FastRelax

FastRelax is designed to optimize the protein backbone/side chains to model at an energy minimum

Helps relieve clashes that may introduce artifacts into design



FastRelax

input_files/relax.command:

```
~/rosetta_workshop/rosetta/main/source/bin/relax.default.l  
inuxgccrelease @relax.options -s 4HKX_renum.pdb
```

input_files/relax.options:

```
-linmem_ig 100 # specify memory to store rotamer pair interactions  
-use_input_sc      # Include rotamers from the input structure  
-nstruct 1        # Generate 1 model  
-relax:fast        # Do a small cycle number fast relax  
-relax:constrain_relax_to_start_coords # Add coordinate constraints  
to backbone heavy atoms, based on the input structure.  
-scorefile relax.fasc
```

Single state design

Please open single_state_design/input_files/design.xml

Where should you start looking?

```
<PROTOCOLS>
```

```
  Run the design protocol
```

```
    <Add mover="design" />
```

```
  Calculate interface metrics for the final sequence
```

```
    <Add mover="analyze" />
```

```
</PROTOCOLS>
```

Design movers

Design mover:

```
<PackRotamersMover name="design" scorefxn="REF2015"  
task_operations="ifcl,rrf" />
```

Task Operations:

Include rotamer options from the command line

```
<InitializeFromCommandline name="ifcl" />
```

Design and repack residues based on resfile

```
<ReadResfile name="rrf" filename="4HKX.resfile"/>
```

Design control

Important to see how much improvement designs have over a nondesigned model

Please open input_files/design_control.xml

Design mover:

```
<PackRotamersMover name="design" scorefxn="REF2015"  
task_operations="ifcl,rrf" />
```

Task Operations:

Include rotamer options from the command line
<InitializeFromCommandline name="ifcl" />

Design and repack residues based on resfile
<ReadResfile name="rrf" filename="4HKX_control.resfile" />

Making resfiles

Use the python script located in
[protein_design/scripts/define_interface.py](#)

Calculates residues on each side of the interface
using a side chain cutoff (default 5 Å)

If any atom of a residue is within 5 Å of any atom of
a residue on the opposing chain – it's considered to
be an interface residue

Making resfiles

<code>--side1=SIDE1</code>	the chains that make up one side of the interface (as a string, e.g. 'AB')
<code>--side2=SIDE2</code>	the chains that make up the other side of the interface (as a string, e.g. 'CD')
<code>--nearby_atom_cutoff=NBX_ATOM_CUT</code>	SC distance cutoff to define a residue as part of the interface. If any SC atom from a residue on one side is within this cutoff of a residue on the other side it's considered to be in the interface. Default=5.0
<code>--output=OUTPUT</code>	Output name for resfile
<code>--design-side=DESIGN_SIDE</code>	Side of interface to design - either 1 or 2. Defaults to 1.
<code>--native</code>	Just repack the residues on the side flagged "design side"
<code>--repack</code>	Repack side of the interface not being designed

Analysis metrics

Total score: score of the entire complex

Interface score: score of residues that are at the interface

Binding energy (ddG, dG_separated): difference in energy between the bound and unbound partners

Binding density (dG_separated/dSASAx100): ddG divided by the buried surface area. Prevents a low binding energy by increasing buried surface area.

Analysis movers

```
<InterfaceAnalyzerMover name="analyze" scorefxn="REF2015"  
packstat="0" pack_input="0" pack_separated="1"  
fixedchains="H,L" />
```

packstat: activates packstat calculation; can be slow so it defaults to off

fixedchains: comma-delimited list of chain ids to define a group in the interface.

pack_separated: repack the exposed interfaces when calculating binding energy? Usually a good idea.

pack_input: prepack before separating chains when calculating binding energy?
Useful if these are non-Rosetta inputs

Sequence logo

Useful to quickly see which residues are being designed, and what amino acids are being put there

Made by WebLogo application through
[scripts/design_analysis.py](http://scripts.design_analysis.py)



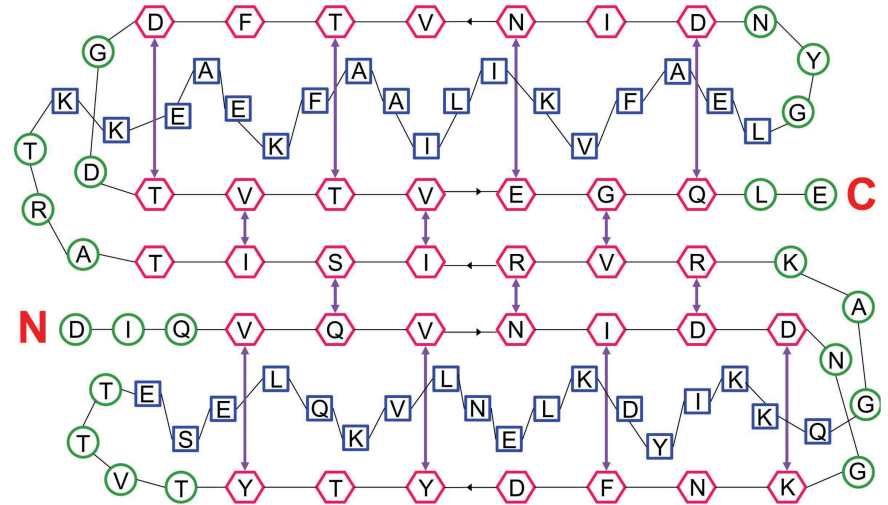
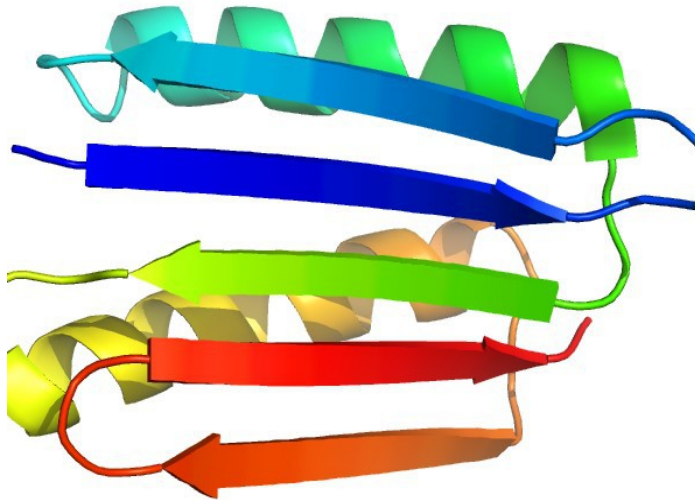
<http://weblogo.berkeley.edu/>

Rosetta Protein Design applications

De Novo Design of a Novel Fold

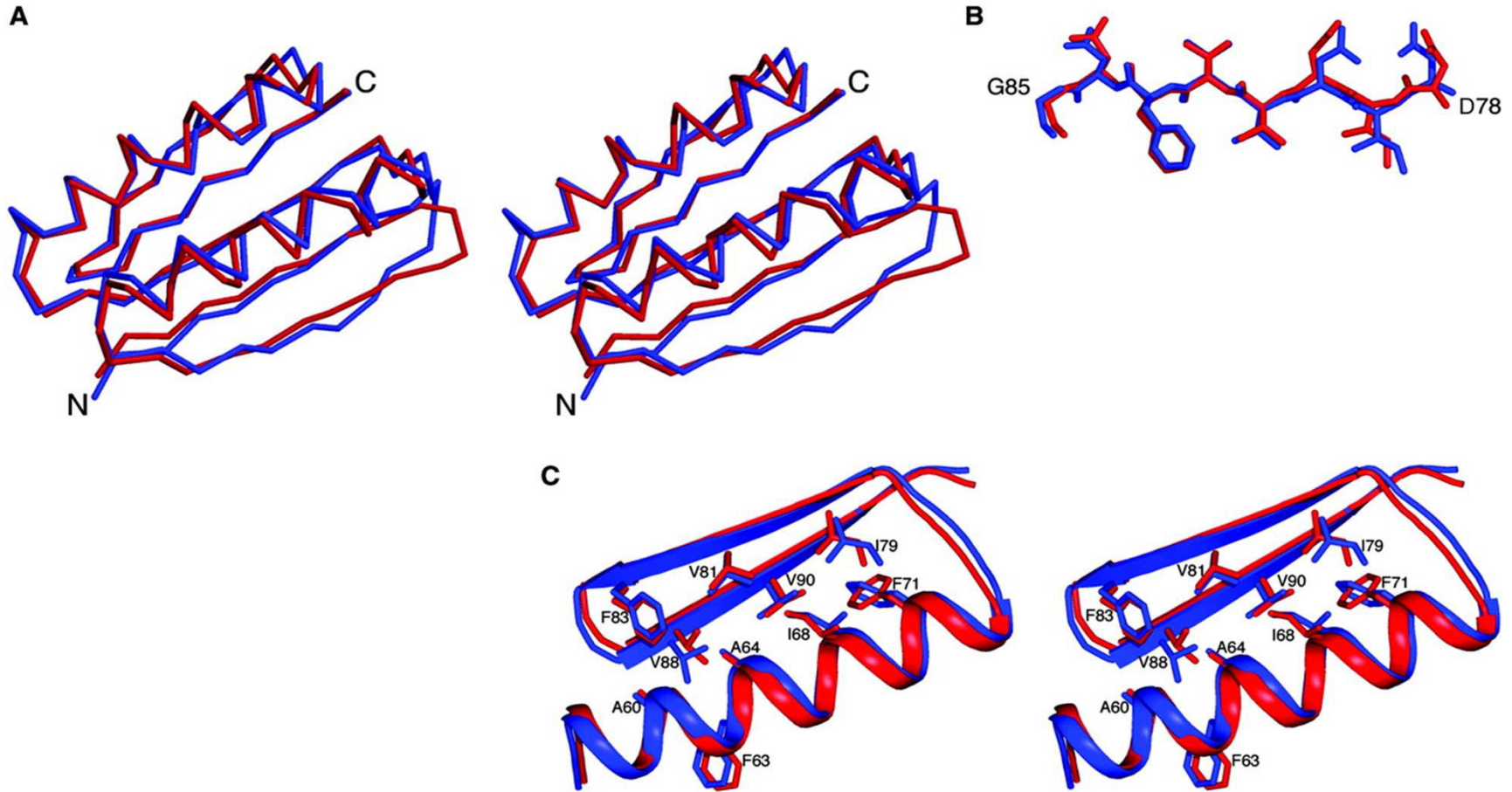
Top7 “back-of-the envelope”
drawn topology not found in the
PDB at time of design

Iterative fixed backbone design +
backbone perturbations



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

Atomic Level Accuracy of Design (blue) to X-ray structure (red)



Additional Design Applications

- Novel Enzyme Design – RosettaMatch and RosettaDesign**

Siegel, J.B. *et al.* (2010). Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction. *Science* 329, 309–313

- Novel Protein Therapeutic Design**

Fleishman, S.J. *et al.* (2011). Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science* 332, 816–821.

- Design of a thermally stabilized enzyme**

Korkegian, A., Black, M.E., Baker, D., and Stoddard, B.L. (2005). Computational thermostabilization of an enzyme. *Science* 308, 857–860.

- Design of self-assembling proteins as nanomaterials**

King, N.P., Sheffler, W., Sawaya, M.R., Vollmar, B.S., Sumida, J.P., Andre, I., Gonen, T., Yeates, T.O., Baker, D. (2012). Computational Design of Self-Assembling Protein Nanomaterials with Atomic Level Accuracy. *Science* 336 1171-1174

Additional Design Applications

- Design of symmetric superfolds to understand protein folding evolution**

Fortenberry, C. *et al.* (2011). Exploring symmetry as an avenue to the computational design of large protein domains. *J. Am. Chem. Soc.* *133*, 18026–18029.

- Rational epitope design**

Wu, X., et al. (2010). Rational design of envelope identifies broadly neutralizing human monoclonal antibodies to HIV-1. *Science* *329*, 856– 861.

- Rational vaccine design**

Jardine, J., et al. (2013). Rational HIV Immunogen Design to Target Specific Germline B Cell Receptors. *Science*.