Antibody Design

Samuel Schmitz
Rosetta Workshop
April 26, 2017

Samuel.Schmitz@vanderbilt.edu

VANDERBILT UNIVERSITY
MEDICAL CENTER

CSB

Overview

Introduction to Protein Design

How it works:

- The Packer
- Resfiles

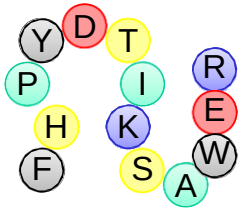
Overview of the Tutorial:

- Input files
- Protocol
- Analysis

Rosetta Design Applications:

- Novel Folds
- Protein-Ligand interactions

Protein Design is the Inverse Protein Folding Problem



Protein Folding Problem

Sequence ⇌ Structure

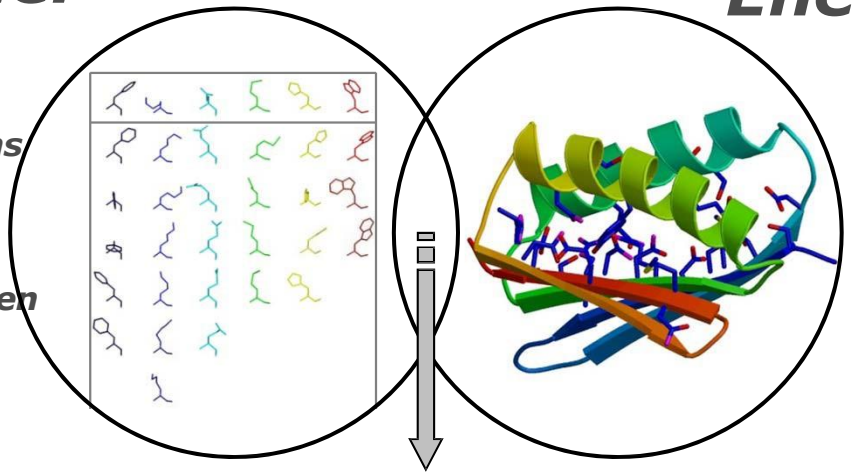
Inverse Folding Problem

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

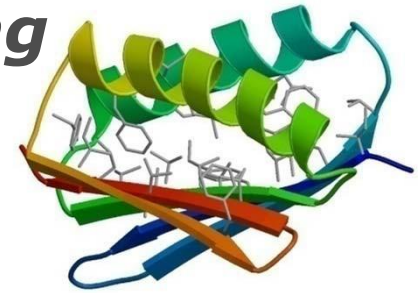
Approximate interactions within sidechain using the distribution of sidechain confor-mations (rotamers) seen in known protein structures



Energy function

Statistically derived potential function
- ***VDW interaction***
- ***solvation***
- ***hydrogen bonding potential***
- ***pair wise interactions***
- ***rotamer probability***

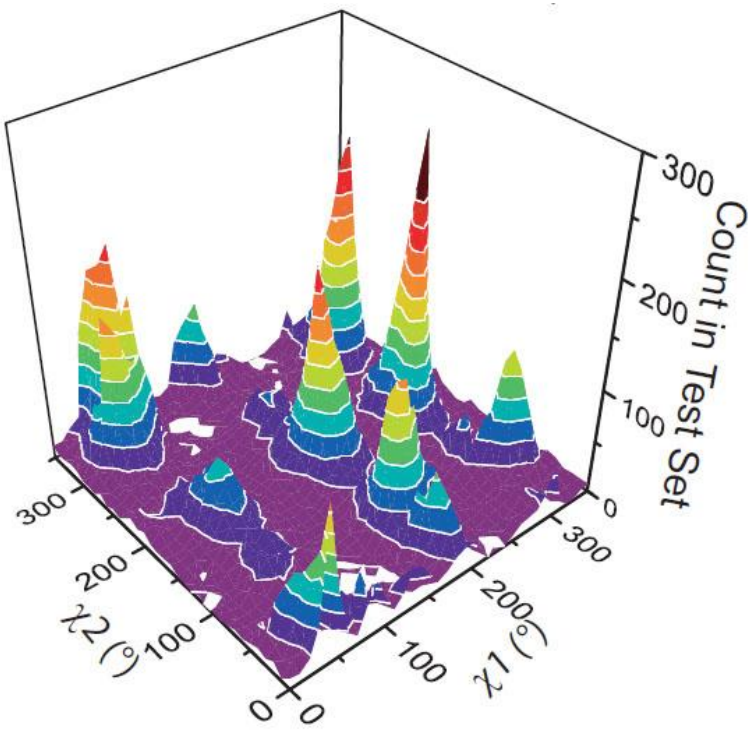
Simulated Annealing Monte Carlo optimization

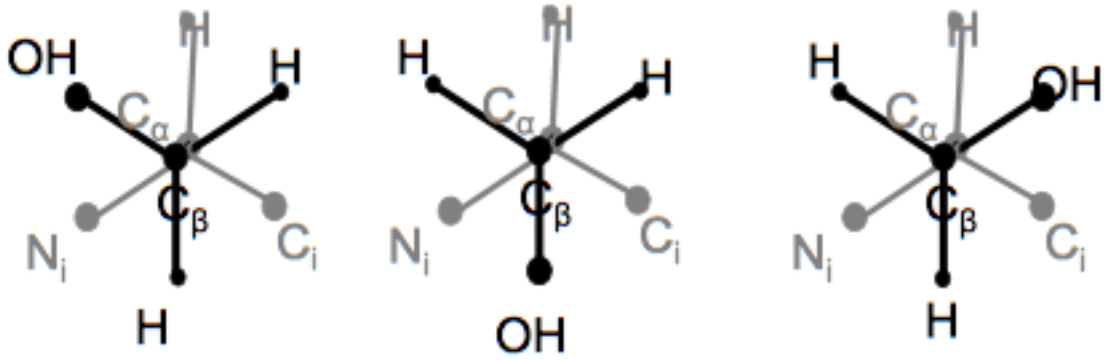


How Rosetta Design works…

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





Adapted from J. Meiler

“The Packer”

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

- Native (repack)
- All Amino Acids (design)
- Specified Amino Acids (guided design)

Steps of the packer:

1. Detect neighbors
2. Build rotamers
3. Calculate energies
4. Simulated annealing/MC Accept

Slide adapted from A. Leaver-Fay

The Packer-What it's Actually Doing

1. Random rotamer substitution
   - Set of rotamers to be considered are specified
2. Evaluate energy
3. Accept/Reject
   - Monte Carlo criterion
   - Simulated annealing from high to low temperature
4. 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 <ResidueTypeName> #allow the specific non canonical residue
```

You can also combine commands (see tutorial).

```
25 A POLAR NOTAA K
```

When on separate lines (say at first a range, then at 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/wiki/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
```

For example, a resfile that does nothing:

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

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
START
30 A PIKAA FY
20-35 A ALLAAxc
```

Tutorial Overview – Antibody design in Rosetta

VANDERBILT UNIVERSITY
MEDICAL CENTER

CSB

Tutorial Overview – Antibody design in Rosetta

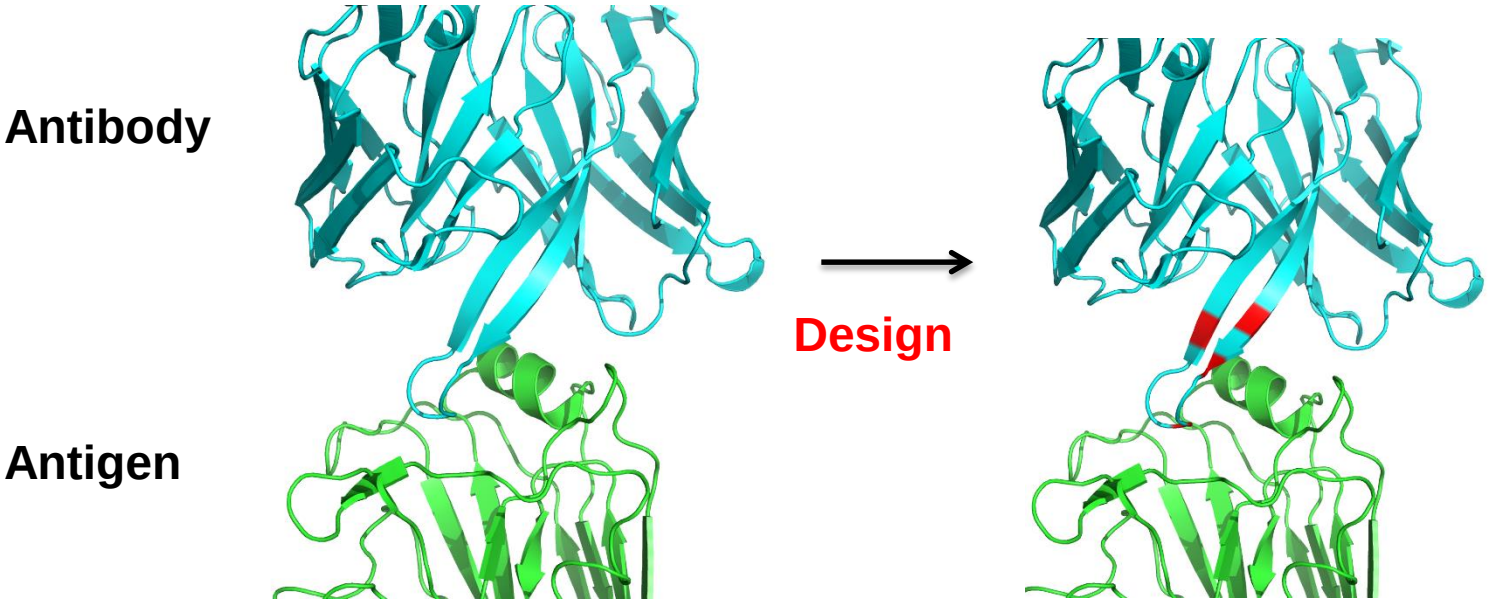
1. Antibody single-state design

2. Antibody multistate design

Antibody single-state design

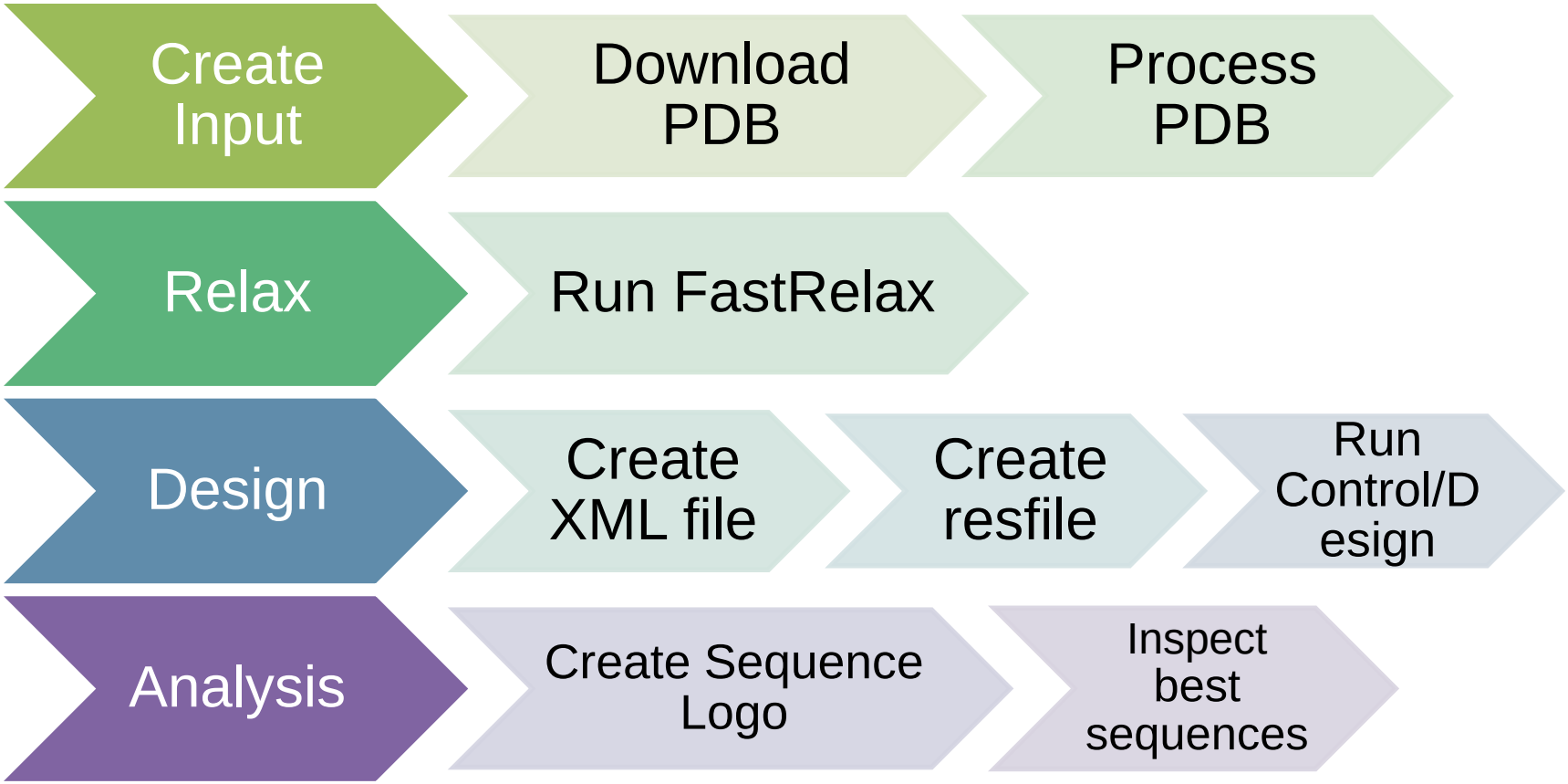
Also known as redesign, computational affinity maturation

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



PDB ID 4yk4

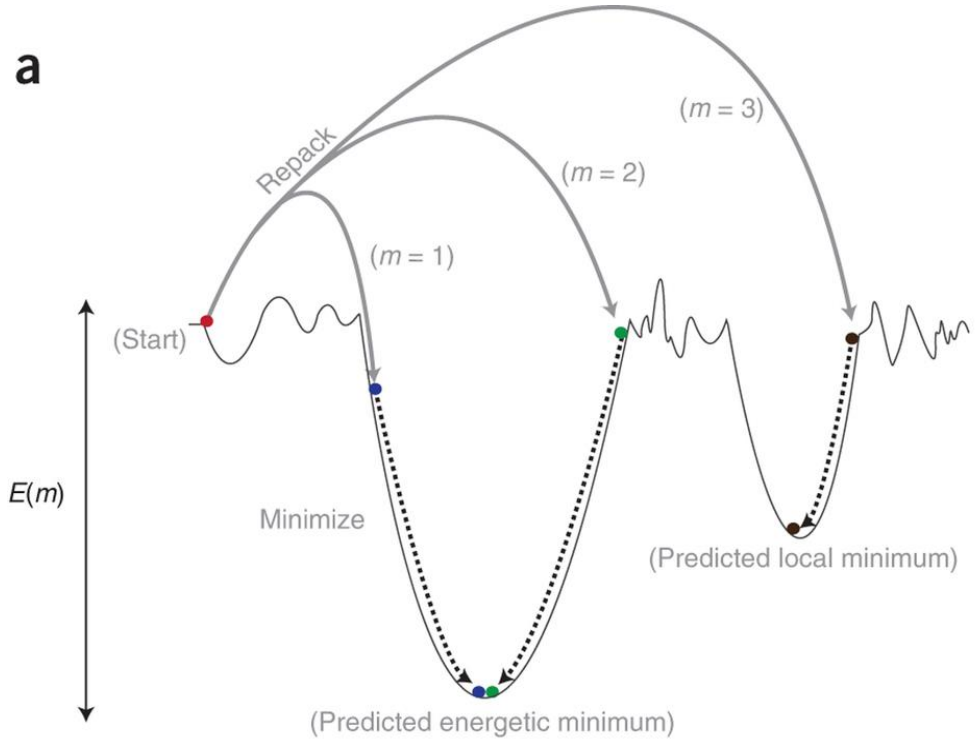
Single-state design protocol overview



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



Combs, *et al*, Nat. Prot. 2013

FastRelax

input_files/relax.command:

```
~/rosetta_workshop/rosetta/main/source/bin/relax.default.linuxgccrelease @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 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=talaris2014
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=talaris2013
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 scripts/define_interface.py

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

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

Making resfiles

```
--side1=SIDE1         # the chains that make up one side of
                      # the interface (as a string, e.g. 'AB')
--side2=SIDE2         # the chains that make up the other side of
                      # the interface (as a string, e.g. 'CD')
--nearby_atom_cutoff=NEARBY_ATOM_CUTOFF
                # 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
--output=OUTPUT       # Output name for resfile
--design-side=DESIGN_SIDE
        # Side of interface to design - either 1 or 2. Defaults to 1.
--native      # Just repack the residues on the side flagged "design side”
--repack      # 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=talaris2014
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 design_analysis.py

http://weblogo.berkeley.edu/

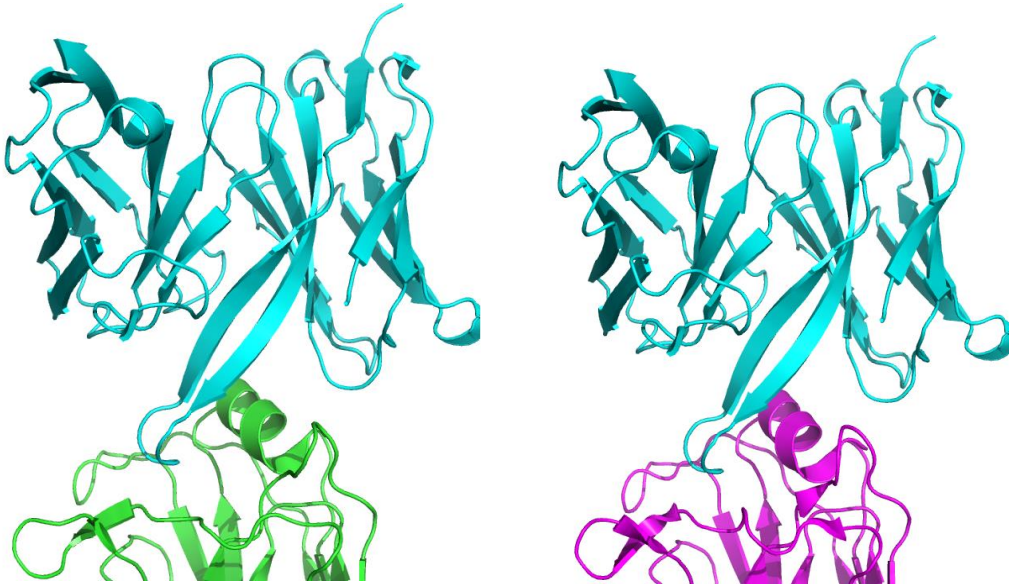
Antibody multistate design

Multistate design: optimize a sequence for low energy in multiple conformations (states)

Redesign an antibody to recognize multiple targets

Antibody

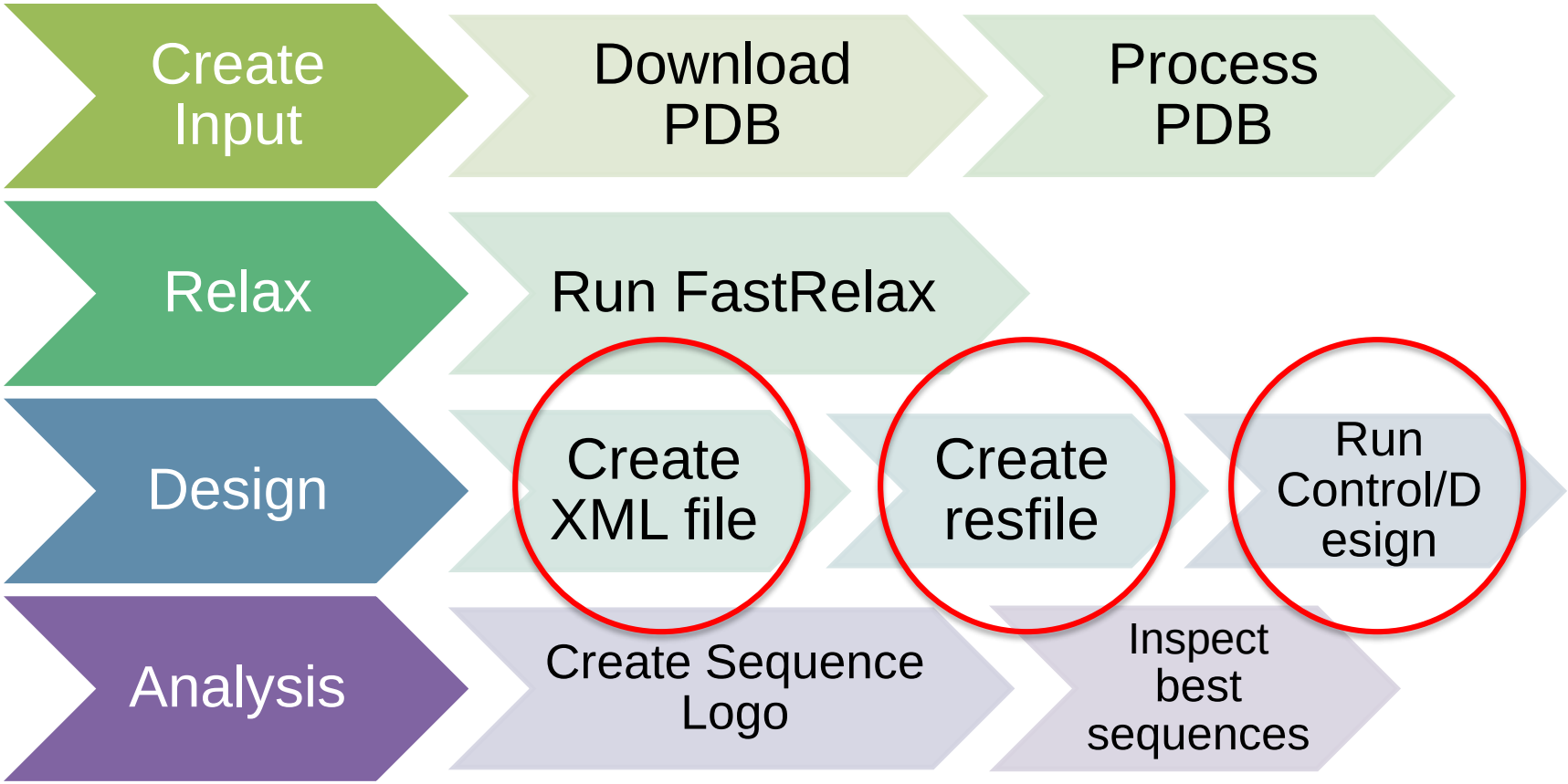
Antigen 1



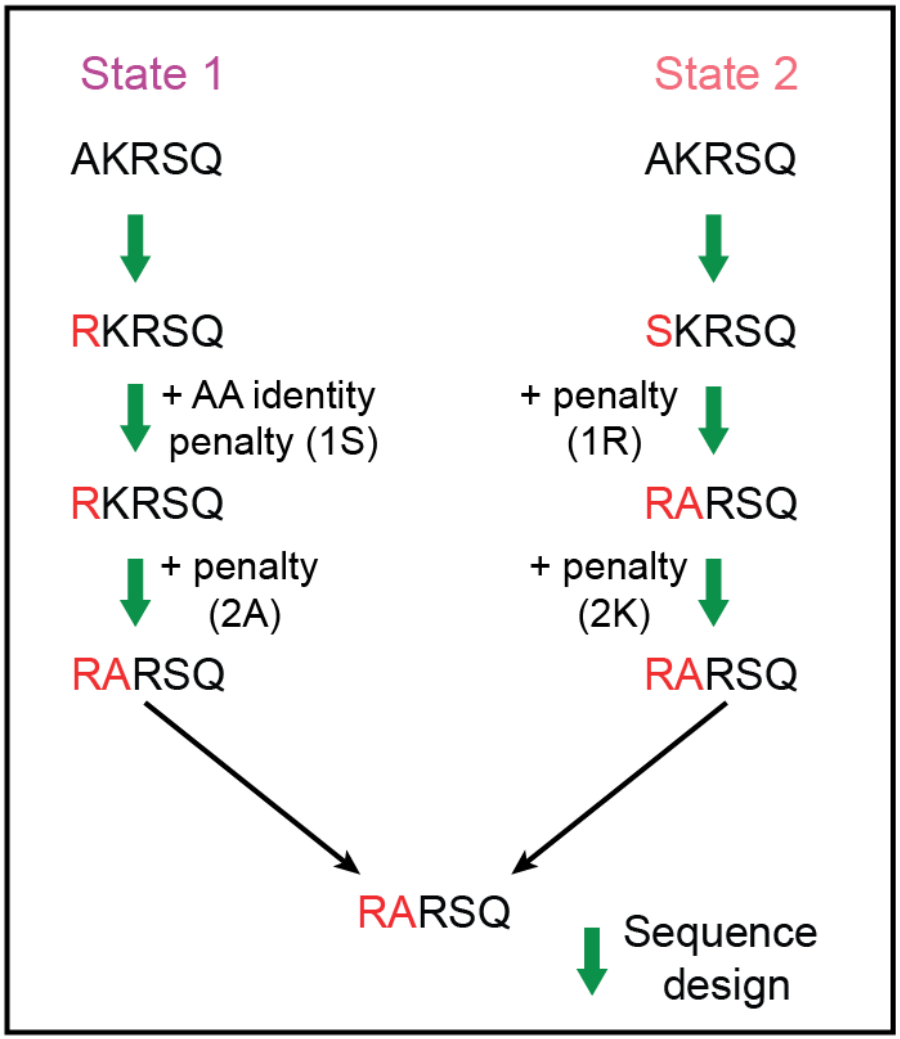
Antigen 2

PDB ID 4yk4

Multistate design protocol overview



REstrained CONvergence in MSD (RECON)



Sevy, A. M., Jacobs, T. M., Crowe, J. E. & Meiler, J. *PLoS Comput. Biol.* **11,** e1004300 (2015).

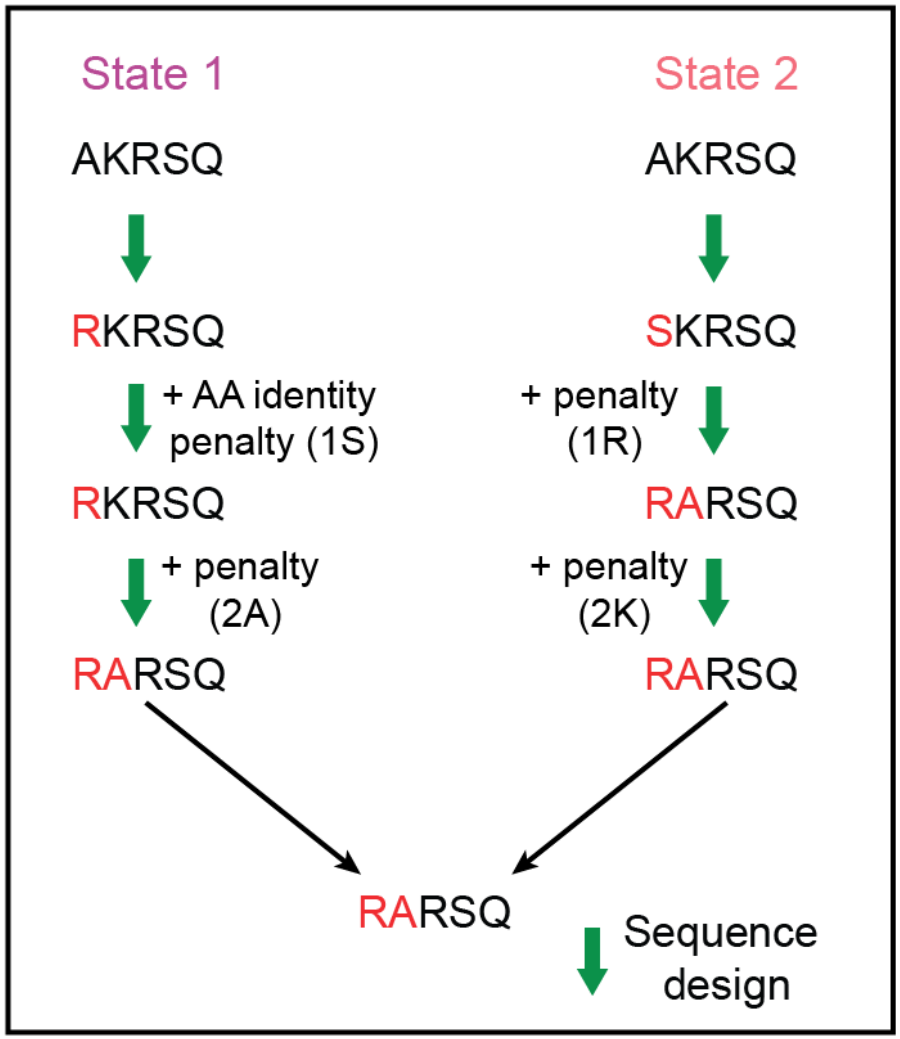
REstrained CONvergence in MSD (RECON)

MSDMover 1

MSDMover 2

MSDMover 3

FindConsensusSequence



Sevy, A. M., Jacobs, T. M., Crowe, J. E. & Meiler, J. *PLoS Comput. Biol.* **11,** e1004300 (2015).

Multistate design protocol

```
<PROTOCOLS>
	Run four rounds of design
	<Add mover=msd1 />
	<Add mover=msd2 />
	<Add mover=msd3 />
	<Add mover=msd4 />

	Find a consensus sequence for all states
	<Add mover=finish />

	Calculate interface metrics for the final sequence
	<Add mover=analyze />
</PROTOCOLS>
```

Multistate design movers

```
<PackRotamersMover name=design scorefxn=talaris_cst
task_operations=ifcl />

<MSDMover name=msd1 design_mover=design
constraint_weight=0.5 resfiles=4HKX.resfile,3UBQ.resfile />
<MSDMover name=msd2 design_mover=design
constraint_weight=1.0 resfiles=4HKX.resfile,3UBQ.resfile/>
<MSDMover name=msd3 design_mover=design
constraint_weight=1.5 resfiles=4HKX.resfile,3UBQ.resfile />
<MSDMover name=msd4 design_mover=design
constraint_weight=2.0 resfiles=4HKX.resfile,3UBQ.resfile />


<FindConsensusSequence name=finish scorefxn=talaris_cst
resfiles=4HKX.resfile,3UBQ.resfile />
```

Multistate design movers

```
<SCOREFXNS>
       <talaris_cst weights=talaris2014.wts >
          <Reweight scoretype=res_type_constraint weight=1.0 />
       </talaris_cst>
</SCOREFXNS>
```

Have to reweight res_type_constraint term to allow for residue constraints!

If it’s not turned on protocol will run but will give a warning

Multistate design tips

You can use multiple resfiles – lets you be more flexible in which residues are being designed/repacked

Resfiles are matched to structure by order of input – **make sure these are in the same order!**

multistate_design.xml:

```
<MSDMover name=msd1 design_mover=design
constraint_weight=0.5 resfiles=4HKX.resfile,3UBQ.resfile />
```

multistate_design.options:

```
-s 4HKX_relax.pdb 3UBQ_relax.pdb
```

Cysteine design is not recommended

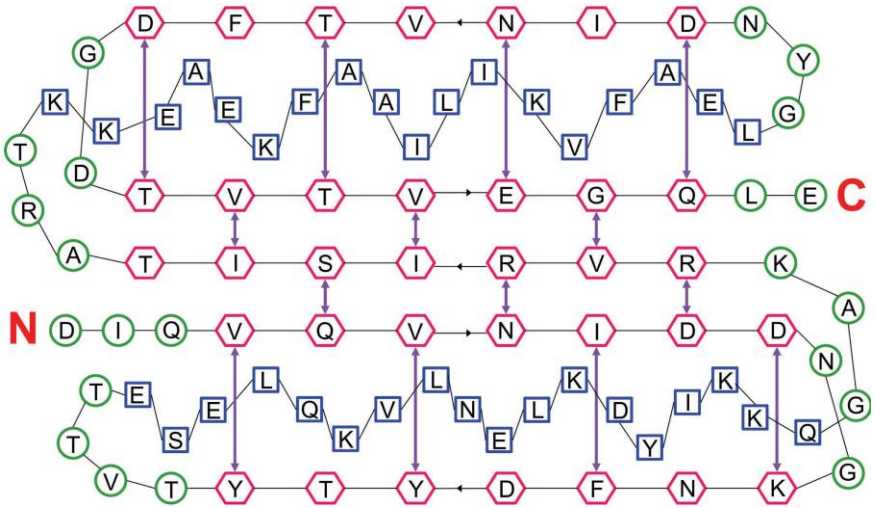
Make sure all resfiles have same number of residues being designed!

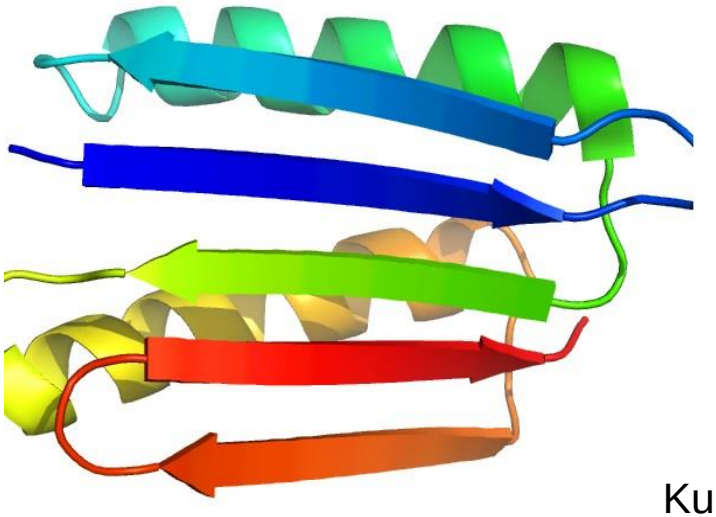
Rosetta Protein Design applications

De Novo Design of a Novel Fold

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

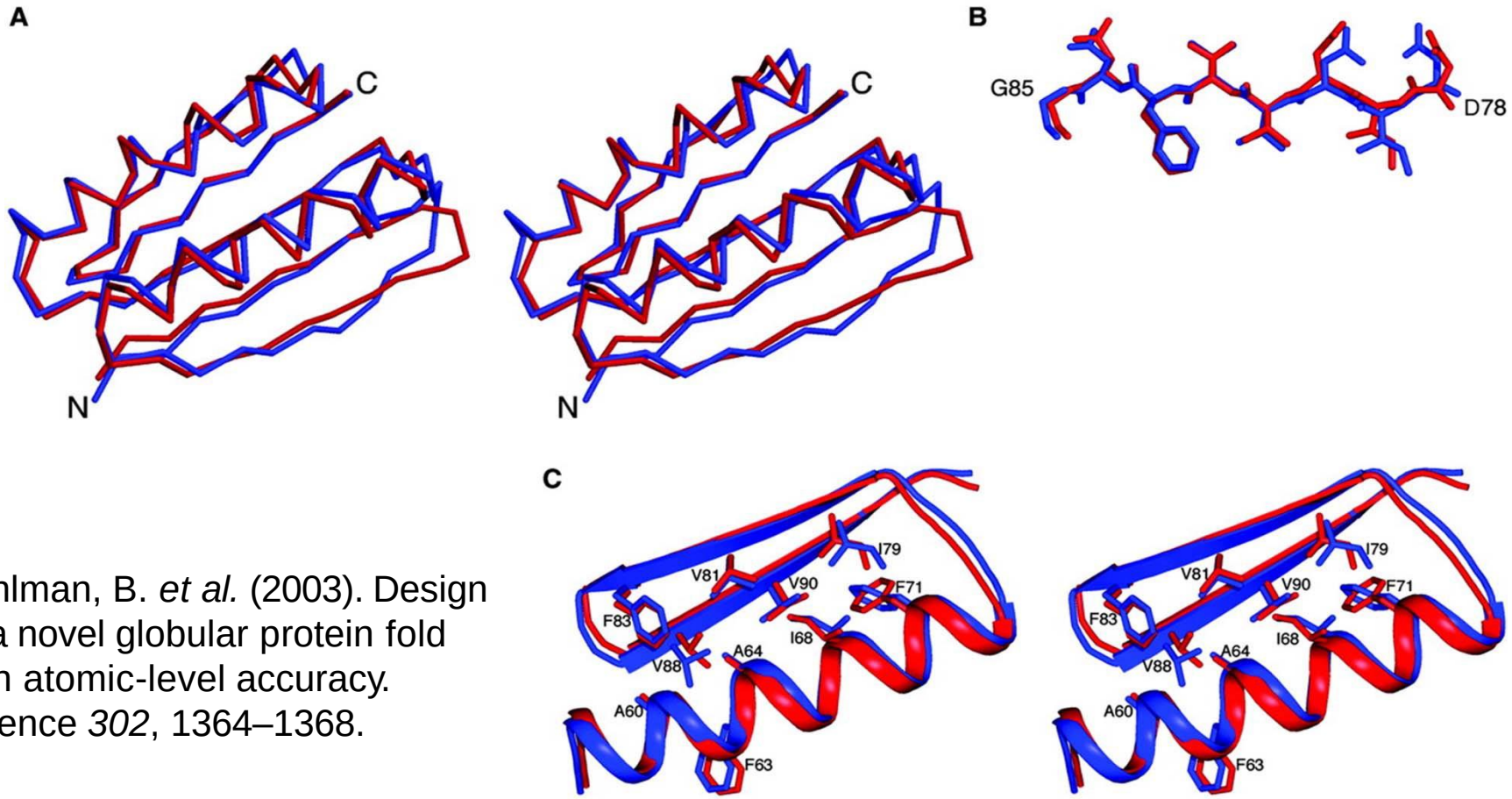
Iterative fixed backbone design + backbone perturbations





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)

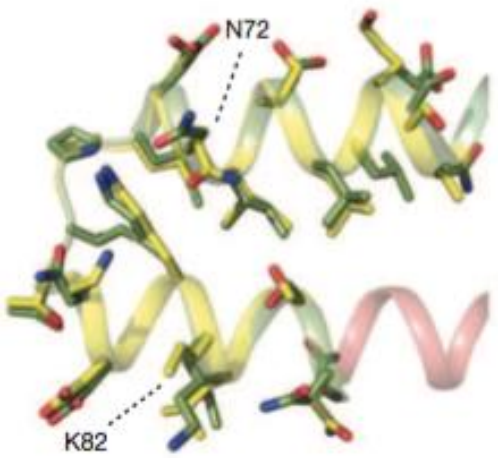


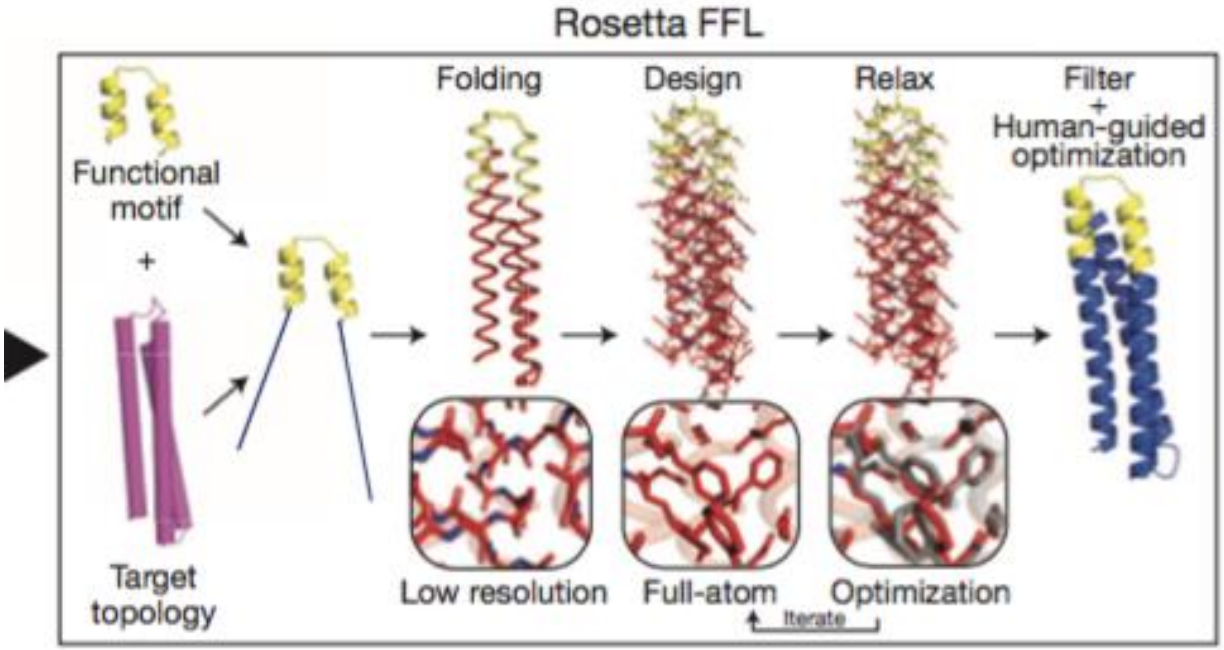
Kuhlman, B. *et al.* (2003). Design of a novel globular protein fold with atomic-level accuracy. Science *302*, 1364–1368.

Design of epitope scaffolds

Extract a known neutralizing epitope from an antigen, place onto a scaffold protein

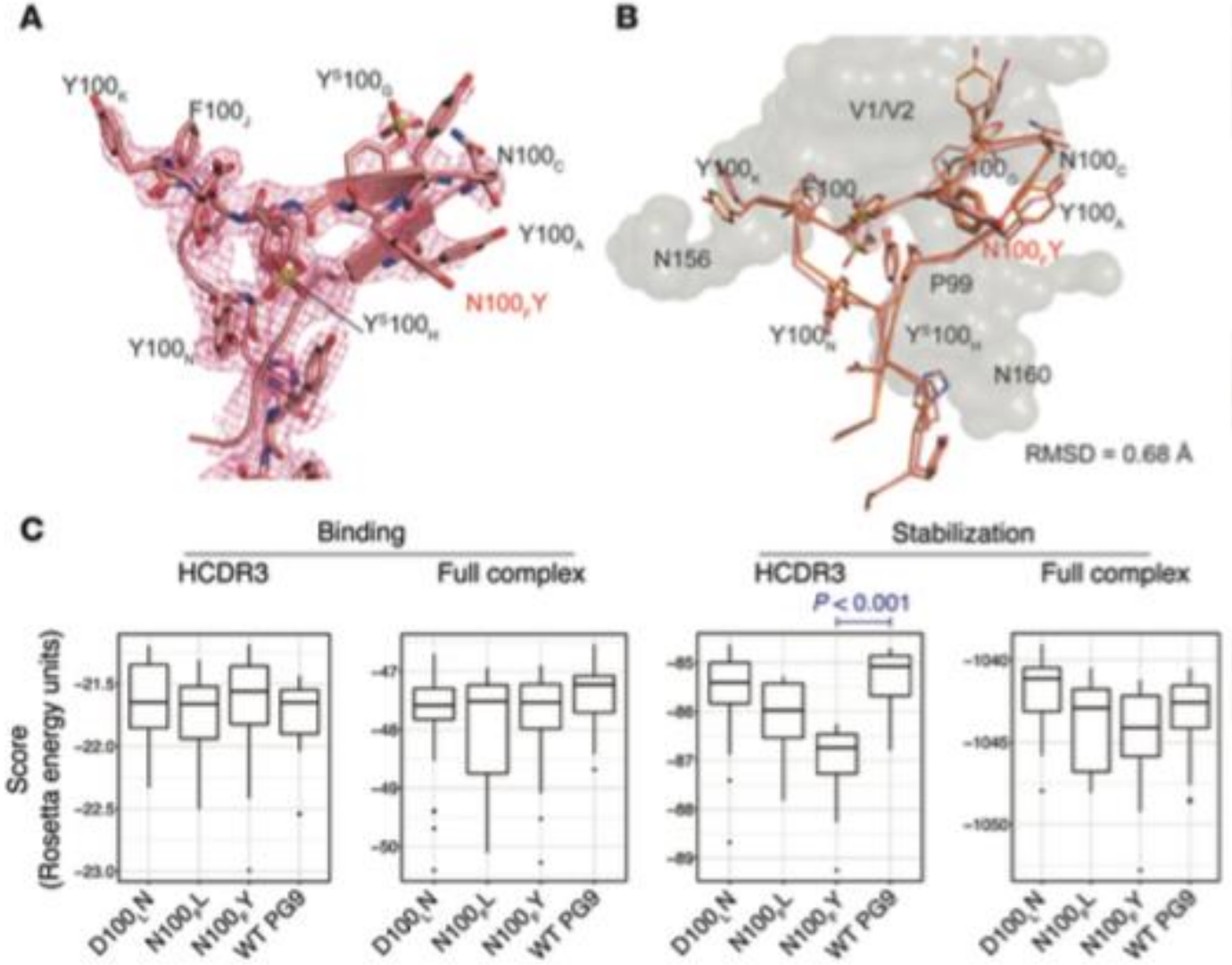
Fold a helix-loop-helix motif, redesign sequence to increase stability





Correia, B. E. *et al.* Proof of principle for epitope-focused vaccine design. *Nature* **507,** 201–206 (2015).12-216

Redesign of HIV antibody with increased potency



Willis, J. R. *et al.* Redesigned HIV antibodies exhibit enhanced neutralizing potency and breadth. *J. Clin. Invest.* **125,** 2523–2531 (2015).

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.