**Mes-binding protein design protocol**

**Part 1 Constructing protein-ligand complex**

1. Manually place mes-acr ligand into enzyme-binding pocket of 7AUY structure using the native ligand as a guide in pymol.
2. Delete native ligand and use pymol to combine new ligand and enzyme into 1 complex using the following commands in pymol
   1. Renames of molecules to the same chain
      1. alter 7auy, chain='A'
      2. alter mes-acr, chain=’A’
   2. pdb convention states ligand should be after main protein atoms, so increase the residue numbering of ligand
      1. alter mes-acr, resv+=500 (I used 500 because 7AUY is about 400 residues)
   3. clear any selection and sort
      1. alter all, segi=''
      2. sort
   4. Join both molecules into 1 protein-ligand complex file and save it
      1. create combo, 7auy or mes-acr
      2. save filename.pdb, combo

Note: step 2 is a lot simpler to do in Chimerax, simply select the two molecules and type the command “combine”.

We can specify sequence of models using the chain IDs as well, since pdb convention is to have the ligand (HETATOM) after the protein sequence. For example, if chain 1 is the ligand and chain 2 is the protein, if we were to join the 2 structures using “combine”, the ligand will appear before the protein residues in the PDB file.

Instead, if we use <combine #2,1 name ‘docked\_structure’>, a new structure named “docked\_structure” will be produced. When saved as a pdb, the ligand will be after the protein. In addition, if the intended complex should only be 1 protein bound to a ligand, remember to change the chain IDs of the new structure, as well as renumber the residues so that the ligand is the last (go to tools -> structure editing)

**Part 2 Protein engineering and using RFDiffusion all atom to generate backbones**

Approach 1: Manually change residues in the binding pocket to interact with mes-acr and allow RFdiffusion to build a structure around it

1. Use pymol to manually mutate residues into those that can interact with mes-acr
   1. Remove charged residue and introduce hydrophobic residue that fits into the structure: His86 -> Met
   2. Introduce residue that can potentially interact with positively charged N in mes-acr: His268 -> Glu
2. Run RFDiffusion with the mutated structure as template, fixing the two residues (Met86 and Glu268) with the command:

*python run\_inference.py inference.deterministic=False diffuser.T=200 inference.output\_prefix=output/ligand\_protein\_motif/mes\_entire\_ inference.input\_pdb=input/design\_1.pdb contigmap.contigs=[\'10-100,A86-86,50-200,A268-268,10-100\'] contigmap.length="150-300" inference.ligand=MES inference.num\_designs=5 inference.design\_startnum=0*

\* design\_1.pdb is the filename of template

\* *'10-100,A86-86,50-200,A268-268,10-100\'* -> this command means to fix residue 86 and 268, then generate 50-200 residues between them. 10-100 residues are also generated at both ends of the protein

1. This allows the model to freely generate an entirely new protein structure around the ligand while ensuring the two introduced residues are not changed.

Approach 2: Same as approach 1 but also fix the two cap sub-domains described in the paper.

**Rationale:** I reasoned that mes-acr is a fairly hydrophobic molecule and a hydrophobic pocket may be generated by the models. The cap domains will then be needed to shield the active site from solvent.

1. Use pymol to manually mutate residues into those that can interact with mes-acr
   1. Remove charged residue and introduce hydrophobic residue that fits into the structure: His86 -> Met
   2. Introduce residue that can potentially interact with positively charged N in mes-acr: His268 -> Glu
2. Run RFDiffusion with the mutated structure (approach 1) as template, fixing the two residues (Met86 and Glu268) **AND** the cap domain residues with the command:

*python run\_inference.py inference.deterministic=False diffuser.T=200 inference.output\_prefix=output/ligand\_protein\_motif/mes\_fixcap- inference.input\_pdb=input/design\_1.pdb contigmap.contigs=[\'A2-43, 5-50, A86-86, 50-200, A172-225, 5-50, A268-268, 5-50\'] contigmap.length="150-350" inference.ligand=MES inference.num\_designs=5 inference.design\_startnum=0*

1. This allows the model to freely generate an entirely new protein structure around the ligand while ensuring the two introduced residues and cap domain residues are not affected.

Approach 3: Only redesign binding pocket of the enzyme

1. Identify resides in the binding pocket:
   1. Residues described in the paper
   2. Also used pymol to locate residues within 5 angstroms of ligand
2. It may be easier to select these residues in pymol and then printing them with the command:
   1. iterate sele and name CA, print (resi, resn)

This prints the amino acid and number to the console, which you can copy and paste.

1. Residues that were identified in 7AUY:

['13', '27', '31', '75', '76', '77', '80', '81', '86', '143', '144', '202', '267', '268', '269', '270', '272', '273']

1. Run RFDiffusion with the original structure as template, fixing the entire protein with the exception of residues identified in step 3:

*python run\_inference.py inference.deterministic=False diffuser.T=200 inference.output\_prefix=output/ligand\_protein\_motif/mes\_activesite\_ inference.input\_pdb=input/design\_1.pdb contigmap.contigs=[\'A2-12, 2-5, A14-26,2-5, A28-30, A32-74, 2-5, A78-79, 2-5, A82-85, 2-5, A87-142, 2-5, A145-201, 2-5, A203-266\'] contigmap.length="150-350" inference.ligand=MES inference.num\_designs=5 inference.design\_startnum=0*

1. This approach restricts the model only to residues in the binding pocket, so there won’t be drastic differences between the generated structures and the template.

**Part 3 Using Protein MPNN to generate protein sequences and Alphafold to predict structures**

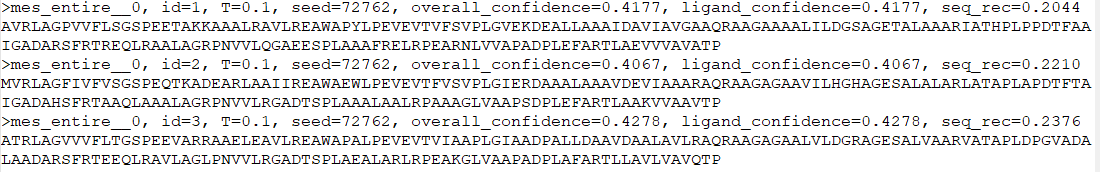
**Rationale**: The heme-binder workflow in the paper used protein MPNN first before Ligand MPNN to generate sequence diversity, so I’m doing the same for the generated backbones all three approaches above.

1. Manually review the structures from RFDiffusion (generated structures) and pick potential candidates.
   1. The main reason for doing this is due to lack of Alphafold 3 compute allocation, so I try to pick structures that are promising rather than all of them.
   2. The decision is of which generated structure to choose is fairly arbitrary but structures that are obviously flawed (steric hindrance in pocket) will not be picked
2. For each potential candidate, look through it and identify the residues introduced in Part 2 (Met86 and Glu268), these will need to be fixed in protein MPNN as well.
   1. Note: Although it shouldn’t have an effect, my experiments suggest that fixing the cap domain residues may be necessary for structures generated in part 2 approach 2. I tend to get sequences whose predicted structures match the template better when I do that.
3. Run protein MPNN with the generated structure as template using the code below. For each structure, I generate a total of 15 sequences in 3 batches:

*python run.py --pdb\_path "./diffused\_1.pdb" --model\_type "protein\_mpnn" --out\_folder "./outputs/mes\_binder" --fixed\_residues "A82 A139" --batch\_size 5 --number\_of\_batches 3*

*\** in this example, the residues Met86 and Glu268 has become residues 82 and 139 in the generated structure whose filename is diffused\_1.pdb

1. Protein MPNN provides confidence scores for each sequence it generates, use that to pick the best sequences for alphafold prediction. It is best to pick several sequences, not just the one with the best score. For example:



In this output, I would choose the sequence with overall\_confidence=0.4278 and 0.4177 to send to alphafold.

1. Once 3D structures are predicted, perform an alignment of the predicted structures against the generated structure that was used as a template in pymol. Pick the ones with the lowest RMSD.
   1. There is a python script to do this automatically that I can send you using pymol installed in Linux.
   2. This is useful when comparing many structures to one template.
2. Because the predicted structures from Alphafold does not contain the ligand, the ligand will need to be manually added before performing Ligand MPNN.
   1. Using the aligned (generated and predicted) structures from step 5
   2. Delete all atoms of the generated structure **saving** the ligand.
   3. Repeat the steps in part 1 to create a protein-ligand structure.

**Part 4 Ligand MPNN and docking**

1. Using the structure from part 3 step 6 as input, run Ligand MPNN with the following command:

*python run.py --pdb\_path "./structure.pdb" --model\_type "ligand\_mpnn" --out\_folder "./outputs/}" --batch\_size 5 --number\_of\_batches 3 --temperature 0.2*

*\** using a higher temperature setting (default is 0.1) generates higher diversity in the protein sequences.

1. Choose sequences based on overall confidence as in part 3 step 4 and send for structure prediction using Alphafold 3.
2. Use predicted structure and ligand for docking.

**Final comments**

I suspect that diffusion is not necessary when using approach 3 since the model doesn’t seem to do much when it is only allowed to act on the small number of residues around the active site.