

De Novo Epitope-Scaffold Design

Bold text means that these files and/or this information is provided.

Italicized text means that this material will NOT be conducted during the workshop

fixed width text means you should type the command into your terminal

If you want to try making files that already exist (e.g., input files), write them to a different directory (`mkdir my_dir`)! Copy & pasting can sometimes lead to weird errors, so when in doubt try to type the commands instead.

Introduction

The membrane-proximal external region (MPER) of the HIV Envelope protein Env is highly conserved, and antibodies against these regions, such as the antibody 10E8, are capable of neutralizing >98% of circulating strains. However, the MPER is sterically occluded in the pre-fusion conformation of native Env, making it difficult to induce antibodies against this region by vaccination with Env. To overcome this challenge, we will design an epitope scaffold that stabilizes the MPER in the 10E8-bound conformation.

To accomplish this, we will combine several methods that you already know from the previous tutorials:

- Generate *de novo* protein backbones using RFDiffusion
- Design amino acid sequences for the scaffold using ProteinMPNN
- Predict the structure of the protein complex using AlphaFold 2 to validate the design.

De novo epitope scaffold design with RFDiffusion

We can generate a protein scaffold for a motif that is known to fulfill a specific purpose. To scaffold protein motifs effectively, we require:

1. A method to specify the desired protein input (one or more segments from a .pdb file)
2. A way to define the connections between these segments in the new scaffolded protein
3. The ability to specify the number of residues for these connections

Additionally, we need to sample various connection lengths, as the optimal number of residues for scaffolding a motif is often not known in advance. Change to the directory “my_files” and - if it does not already exist - create the last result directory.

```
cd ~/rosetta_workshop/tutorials/scaffolding/my_files/
mkdir -p 1_RFDiffusion 2_MPNN 3_ColabFold
cd 1_RFDiffusion
```

1. In this example we look at the interaction of membrane-proximal external region (MPER) of the HIV Envelope protein Env and antibody 10E8 provided in “input_files/8U08.pdb”. We want to extract MPER and design an epitope scaffold that stabilizes MPER in the 10E8-bound conformation. In this example, we will use RFDiffusion to generate a scaffold that builds 39 residues (“39-39”) N-terminally of “C96-107” from the input pdb, followed by “20-50” residues to its C-terminus. All new parts are randomly sampled at each inference cycle. Importantly, we want to inpaint the presence of the Fv fragments for the heavy and light chain of the 10E8 antibody with “H1-131/0 L2-119/0” to avoid clashes of the newly generated scaffold with the antibody.

```
~/rosetta_workshop/Rfdiffusion/scripts/run_inference.py \
  inference.input_pdb=./../input_files/8U08.pdb \
  'contigmap.contigs=[39-39/C96-107/20-50/0 H1-131/0 L2-119/0]' \
  inference.output_prefix=./1_RFDiffusion/scaffold \
  inference.num_designs=2
```

2. Use PyMOL to investigate your results:

First let's color by chain (*util.cbc*). You will see that only two chains were generated (one for the antigen and one for the antibody). The antibody is technically composed of the heavy and the light chains, but during RFDiffusion backbone generation, the two were merged into a single chain. Next, let's print the protein sequences (*print*). You will notice that the newly designed antigen scaffold (chain A) is almost exclusively composed of glycine residues, with the exception being the portions that stemmed from the template PDB (the MPER and the antibody). Furthermore, if you try to show the sidechain for the full pose (*show stick*), you will see that both the antigen and the antibody are composed only of backbone atoms and none of the sidechains can be displayed. The commands are the following:

```
cd ~/rosetta_workshop/tutorials/scaffolding/my_files/2_MPNN
cp ../../input_files/rfdif_backbone.pdb .
pymol rfdif_backbone.pdb
    util.cbc
    print cmd.get_fastastr('chain A+B')
    show sticks
```

Immunogen design with protein MPNN

2. As you saw in PyMOL, RFDiffusion only designs the backbone, leaving all residues as glycine. Consequently, we need to re-design the backbone generated with RFDiffusion using ProteinMPNN. We will be using ProteinMPNN in this tutorial, and have to activate the corresponding conda environments for each step:

```
conda activate pmpnn_tutorial
```

3. The template contains both the antibody and the epitope-scaffold, but we only want to redesign the epitope scaffold. We therefore restrict design to chain A.

```
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/parse_multiple_chains.py \
    --input_path . --output_path rfdif_backbone.jsonl
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/assign_fixed_chains.py \
    --input_path rfdif_backbone.jsonl --chain_list "A" \
    --output_path rfdif_backbone_assigned.jsonl
```

4. We do not want to redesign the residues of the MPER in direct contact with the antibody. We use pymol to determine which residues to keep constant, assuming that MPER residues (residues 40-51 of chain A) within 7 Angstrom of the antibody form direct contacts.

```
pymol rfdif_backbone.pdb
```

To select the interface and print the residue numbers, we type into the pymol command line:

```
select interface, chain A and resi 40-51 within 7 of chain B
iterate interface and name CA, print (resi)
```

5. We can now set up ProteinMPNN to keep these residues fixed:

```
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/make_fixed_positions_dict.py \
    --position_list "40 41 44 45 47 48 51" --chain_list "A" \
    --input_path rfdif_backbone.jsonl --output_path rfdif_backbone_fixed.jsonl
```

6. Now we design the residues outside the epitope using ProteinMPNN.

```
mkdir Outputs
python ~/rosetta_workshop/ProteinMPNN/protein_mpn_run.py \
    --jsonl_path rfdif_backbone.jsonl --chain_id_jsonl rfdif_backbone_assigned.jsonl \
    --out_folder Outputs --num_seq_per_target 100 --sampling_temp "0.1" \
    --fixed_positions_jsonl rfdif_backbone_fixed.jsonl --batch_size 1
```

The sequence design will take 1-2 minutes to generate 100 designs. It will create a fasta file with the designed sequences in the folder Outputs/seqs/rfdif_backbone.fa. The file contains the original sequence and all designs. The name of each design contains the score, global_score and sequence recovery. The score only includes residues that were designed, whereas the global_score also includes scores from fixed residues. In this first round of design, the sequence recovery is irrelevant, as the starting sequence (except for the MPER) was all glycine. However, we will next filter for the best-scoring designs.

7. We will use a script to extract the 10 best-scoring designs and form the consensus sequence of those designs.

```
cp ../../input_files/calculate_consensus.py .
python calculate_consensus.py Outputs/seqs/rfdif_backbone.fa top_designs_R1.fa
```

8. Next, we want to use Alphafold to predict the structure of the designed sequences. Instead of using alphafold 2 directly, we will use ColabFold to predict the structure, which uses a faster MSA algorithm, reducing computational cost. The script already generated the (consensus) immunogen sequence. However, to predict the structure of the immunogen in complex with the antibody, we still need to extract the sequences of heavy and light chain. We will once again use pymol to extract the sequence:

```
cd ../3_ColabFold/
cp ../../input_files/rfdif_input.pdb .
pymol rfdif_input.pdb
```

To print the sequence of the antibody, we type into the pymol command line:

```
print cmd.get_fastastr('chain H+L')
```

We now combine the sequences of the immunogen, heavy and light chain into one string separated by “:” to indicate new chains, and save it in fasta format. Open a text-editor of your choice and create a file “design_R1.fa” that contains the sequence information, e.g.:

```
gedit design_R1.fa
>R1
MTETEKLEETKKELKSLPEEIRKKVLELLELLKKAGVTWFEELTNILWKVKKLIDEGDEEGLDKFIEEIKEELEKK:EVQL
VESGGGLVKPGGSLRLSCAASGFTFSNAWMSWVRQAPGKGLEWVGRIKSKTEGGTTDYAAPVKGRFTISRDDSKNTLYLQM
NSLKTEDTAVYYCARTGKYDFWSGYPPGEEYFQDWGQGTLVTVSS:ELTQDPAVSVALGQTVRITCQGDSLRSYYASWYQ
QKPGQAPVLLIYGKNNRPSGVPDRFSGSSSGNTASLTITGAQAEDEADYYCNSRDSSGNHLWVFGGGTKLTVL
```

Save the file under “design_R1.fa”. Note that we used the immunogen as the first sequence, which will result in the immunogen becoming chain “A”. If you change the order of the sequences, you will have to adjust the chain IDs in the subsequent steps accordingly.

We also need the instruction file to perform structure prediction:

```
cp ../../input_files/colab_default.job .
```

9. The computers we’re using for this workshop can’t run ColabFold. We will need to run these calculation on a remote cluster (ACCRE, for this workshop). First, copy the 3_ColabFold directory into the workshop folder directory you created previously (replace USERNAME with your ACCRE username):

```
cd ../
rsync -avz 3_ColabFold USERNAME@login.accre.vanderbilt.edu:~/workshop/
```

Next log into ACCRE by running the following command (replace USERNAME and type your password):

```
ssh USERNAME@login.accre.vanderbilt.edu
```

Navigate to the 3_ColabFold directory you just copied and run the slurm script using the following command:

```
sbatch colab_default.job
```

You can check the status of your job on ACCRE with the following command:

```
squeue -u USERNAME
```

The structure prediction will take around 15 min to complete. Once your job is complete, you will need to copy the output back to your local directory. Navigate to the my_files directory on your local machine and run the following command:

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/3_ColabFold 3_ColabFold/
```

10. We next want to filter results for high-quality predictions. The quality scores can be found in the file “log.txt” in the R1_designs. To find the scores of the best model, we can search this file for the keyword “rank_00”:

```
grep rank_00 Outputs/log.txt
```

Typically, we filter designs for pLDDT values higher than 90 and pTM values greater than 0.85. How many good models do you have based on these cut-offs?

11. We also want to ensure that the antibody is predicted to interact directly with the MPER. We will use pymol to calculate the RMSD between the original input structure and the colabfold prediction. First, we copy the best-scoring design to a new folder R2_input:

```
cp Outputs/R1_relaxed_rank_001_*.pdb ./R1_prediction.pdb
```

We then open the original template and the new prediction in pymol:

```
pymol *.pdb
```

We now want to align the structures on the antibody but measure the RMSD of the grafted MPER. First let's align the structures by typing into the pymol command line:

```
super R1_prediction and chain B+C, rf dif_input and chain H+L
```

To measure the RMSD between the MPER regions of both proteins, we will use the “rms_cur” command. However, this function requires the residues of the two MPERs to have the same chain ID and numbering, so we have to renumber the epitope of the template. The template MPER starts at residue 96 of chain C, whereas the MPER of our design starts at position 40 of chain A, so we have to adjust the numbering by 56 and change the chain ID:

```
alter rf dif_input and chain C, resi=int(resi)-56  
alter rf dif_input chain C, chain='A'
```

now we can calculate the RMSD:

```
print cmd.rms_cur('rf dif_input and chain A', 'R1_prediction and chain A')
```

For epitope-scaffolding applications, we typically allow only small RMSDs of lower than 3Å. However, for this tutorial, we will continue if the RMSD is lower than 5Å, otherwise you can continue with the structure R1_prediction.pdb located in the input_files folder of the tutorial.

12. We have now completed one iteration of the design pipeline. To further improve the designs, we can iterate until ProteinMPNN design and AlphaFold structure prediction converge. Therefore, we perform ProteinMPNN design on the new structure, using similar instructions as in steps 3-6:

```
cd ../
mkdir 4_Second_Round
cd 4_Second_Round
cp ../3_ColabFold/R1_prediction.pdb .
```

```
python ~/rosetta_workshop/ProteinMPNN/helper_scripts/parse_multiple_chains.py \
    --input_path . --output_path rfdif_backbone_R2.jsonl

python ~/rosetta_workshop/ProteinMPNN/helper_scripts/assign_fixed_chains.py \
    --input_path rfdif_backbone_R2.jsonl --chain_list "A" \
    --output_path rfdif_backbone_assigned_R2.jsonl

python ~/rosetta_workshop/ProteinMPNN/helper_scripts/make_fixed_positions_dict.py \
    --position_list "40 41 44 45 47 48 51" --chain_list "A" \
    --input_path rfdif_backbone_R2.jsonl --output_path rfdif_backbone_fixed_R2.jsonl

mkdir Outputs

python ~/rosetta_workshop/ProteinMPNN/protein_mpn_run.py \
    --jsonl_path rfdif_backbone_R2.jsonl --chain_id_jsonl rfdif_backbone_assigned_R2.jsonl \
    --out_folder Outputs --num_seq_per_target 100 --sampling_temp "0.1" \
    --fixed_positions_jsonl rfdif_backbone_fixed_R2.jsonl --batch_size 1
```

In contrast to the first iteration, which was initialized from a structure that contained only glycine amino acids, the seq_recovery of the designed sequences in R2_designs/seqs/R1_prediction.fa indicates how well ProteinMPNN and ColabFold have converged. We typically continue iterative structure prediction and sequence design until seq_recovery is >0.95. This is usually achieved after 2-3 iterations. Note: Due to time- and computational constraints, we have only designed a single epitope-scaffold today and the chance of it passing all filters (pLDDT, pTM, RMSD and seq_recovery) is quite low. For a real-world design challenge, we would run 1,000-10,000 design trajectories, varying the size of the scaffold and position of the epitope within the scaffold.

13. Feel free to continue with the ColabFold step of the second round on design on your own!
14. Additional information about protein design with RFDiffusion, ProteinMPNN and AlphaFold: Nathaniel R. Bennett, Brian Coventry, et al. Improving de novo protein binder design with deep learning. Nature communications 2023 May 6;14(1):2625.