Immunogen design in Rosetta

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.

Initial set-up

Immunogen design with protein MPNN

In the current workshop (2024), we have learned how to predict the structure of protein complexes from their sequence using AlphaFold2 (Tutorial 1), to design an aminoacid sequence that matches a given protein backbones using ProteinMPNN (Tutorial 2) and to generate *de novo* protein backbones using RFDiffusion (Tutorial 3). Now we will combine these methods and to apply them to a real-world problem: the design of immunogen candidates against HIV.

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.

1. Create a directory in the tutorial directory named my_files and switch to that directory. We will work from this directory for the rest of the tutorial.

```
mkdir my_files
cd my_files
mkdir 1_MPNN 2_ColabFold
```

2. In Tutorial 3, you have already used RFDiffusion to design a *de novo* protein backbone that stabilizes the MPER in the 10E8-bound conformation, but RFDiffusion only designs the backbone, leaving all residues as glycine. In this first step we will re-design the backbone generated with RFDiffusion using Protein MPNN. Let's copy the backbone into the folder and give a look at it. First let's color by chain (*util.cbc*). You will see that only two chains were generate (one for the antigen and one for the antibody). The antibody is technically composed by the heavy and the light chains, but for simplicity the two were conisered as a single chain during RFDiffusion. Next, let's print out the protein sequences (*print*). As mentioned, the newly designed antigen scaffold (chain A) is composed by glycines everywhere else than the interface. Not only, 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 by backbone atoms and none of the sidechains can be displayed. The commands are the following:

```
cd 1_MPNN
cp ../../input_files/rfdif_backbone.pdb .
pymol rfdif_backbone.pdb
        util.cbc
        print cmd.get_fastastr('chain A+B')
        show sticks
```

3. We will be using ProteinMPNN in this tutorial, and have to activate the corresponding conda environments for each step:

```
conda activate pmpnn_tutorial
```

4. 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
```

5. 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)
```

6. 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
```

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

```
mkdir Outputs
python ~/rosetta_workshop/ProteinMPNN/protein_mpnn_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 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 ../2_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
MTETEKLIEETKKELKSLPEEIRKKVLELLELLKKAGVTWFELTNILWKVKKLIDEGDEEGLDKFIEEIKEELEKK:EVQL
VESGGGLVKPGGSLRLSCAASGFTFSNAWMSWVRQAPGKGLEWVGRIKSKTEGGTTDYAAPVKGRFTISRDDSKNTLYLQM
NSLKTEDTAVYYCARTGKYYDFWSGYPPGEEYFQDWGQGTLVTVSS:ELTQDPAVSVALGQTVRITCQGDSLRSYYASWYQ
QKPGQAPVLLIYGKNNRPSGVPDRFSGSSSGNTASLTITGAQAEDEADYYCNSRDSSGNHLWVFGGGTKLTVL
```

Save the file under "design_R1.fa". Note that we used the immunogen as the first sequence, which will result the immunogen in 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. We will need to run these calculation on ACCRE, as done for Tutorial 1. First, copy the 2_ColabFold directory into the workshop folder directory you created yesterday (replace USERNAME with your ACCRE username):

```
cd ../
rsync -avz 2_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 2_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_file directory on your local machine and run the following command:

```
rsync -avz USERNAME@login.accre.vanderbilt.edu:~/workshop/2_ColabFold 2_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 theMPER. 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, rfdif_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 rfdif_input and chain C, resi=int(resi)-56
alter rfdif_input chain C, chain='A'
```

now we can calculate the RMSD:

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

For epitope-scaffolding applications, we typically allow only small RMSDs of lower than 3A. However, for this tutorial, we will continue if the RMSD is lower than 5A, 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. We therefore perform ProteinMPNN design on the new structure, using similar instructions as in steps 3-6:

```
cd ../
mkdir 3_Second_Round
cd 3_Second_Round
cp ../2_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_mpnn_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 aminoacids, 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 on 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.