RFantibody

This tutorial is based on the 2026 tutorial from DéJenaé See and Riti Biswas

Introduction

In this tutorial, we're going to walk through how to run RFantibody — a pipeline for de novo antibody design.

The pipeline follows three main steps: first, we use RFdiffusion to generate antibody backbone structures, then we use ProteinMPNN to design sequences for those backbones, and finally we use an ML prediction pipeline (e.g. AlphaFold3 or RosettaFold3) to predict the structure of our designs and evaluate them.

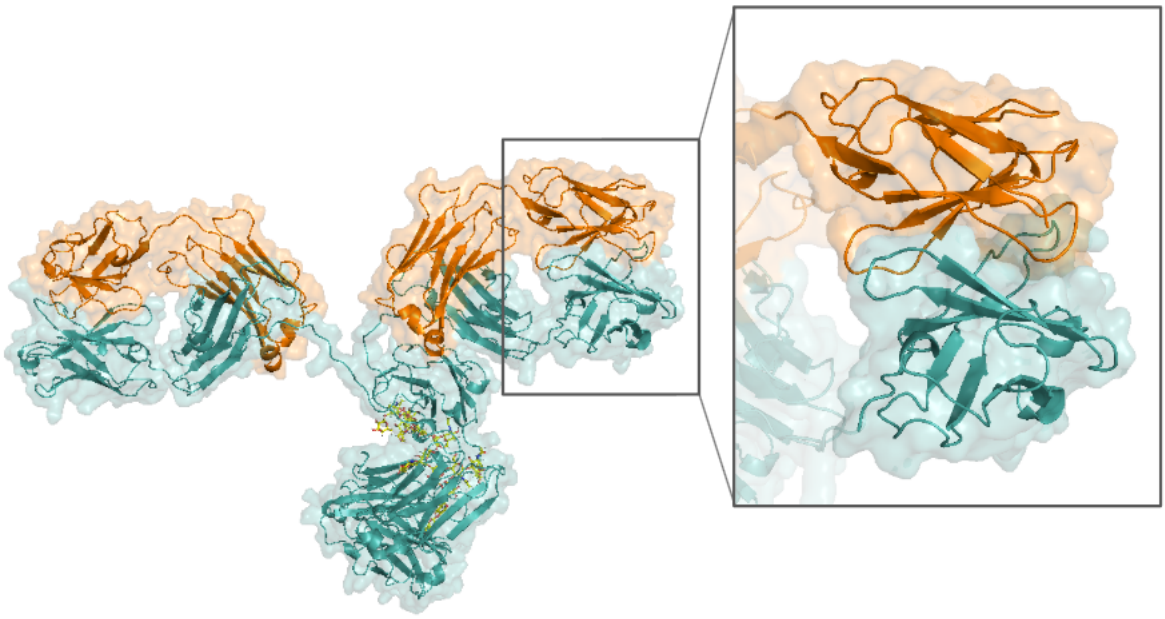
Before getting started: For your own projects, we recommend running RFantibody on GPU nodes within a high-performance computing (HPC) cluster. See the RFantibody README for instructions for installing RFantibody.

Limitations A few other things to keep in mind:

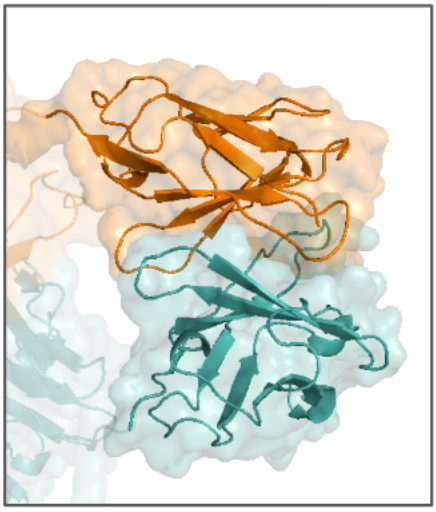
- The model is fine-tuned specifically to design the **complementarity-determining region (CDRs)** of antibodies. It's not trained to design frameworks or other constant regions.
- It was trained on Chothia-defined CDRs — if you define your CDRs by a different numbering scheme, results may not be optimal.
- This version only supports amino acid targets. All-atom design isn't supported yet–though it's coming soon!
- If you're using the AlphaFold3 web server for the prediction step, DeepMind specifies a 5,000 amino acid limit — that's for the combined length of your antibody and antigen.

Background: Antibody structure

Conventional antibodies contain 4 chains–2 heavy chains and 2 light chains. Each heavy chain (teal) forms a dimer with a light chain (orange), as shown below:

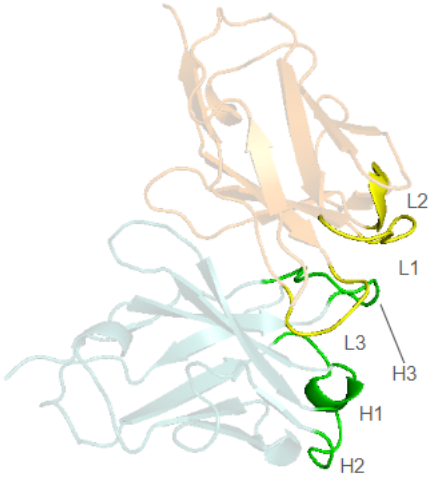


Each chain contains multiple domains. The light chain has one constant domain and one variable domain, VL. The heavy chain contains 3-4 constant domains (depending on the antibody class) and one variable domain, VH. Each variable domain contains three complementarity-determining regions, or CDRs. The CDRs are primarily responsible for antibody binding, and these are the regions that RFantibody is trained to design.



DIVLTQSPSSLSASLGDTITITCHASQNINVWLSWYQQK
PGNIPKLLIYKASNLHTGVPSRFSGSGSGTGFTLTISSL
QPEDIATYYCQQGQSYPLTFGGGTKLEIKRADAAPTVSI
FPPSSEQLTSGGASVVCFLNNFYPKDINVKWKIDGSER
QNGVLNSWTDQDSKDSTYSMSSTLTLTKDEYERHNSY
TCEATHKTSTSPIVKSFNRNEC

EVKLQESGGGLVQPGGSLKLSCATSGFTFSDYYMYW
VRQTPEKRLEWVAYISNGGGSTYYPDTVKGRFTISRDN
AKNTLYLQMSRLKSEDTAMYYCARHGGYYAMDYWGQ
GTTVTVSSAKTTAPSVYPLAPVCGDTTGSSVTLGCLVK
GYFPEPVTLTWNSGSLSSGVHTFPAVLQSDLYTLSSSVT
VTSSTWPSQSITCNVAHPASSTKVDKKIEPRGPTIKPCP
PCKCPAPNLLGGPSVFIFPPKIKDVLMISLSPIVTCVVVD
VSEDDPDVQISWFVNNVEVHTAQTQTHREDYNSTLRV
VSALPIQHQDWMSGKEFKCKVNNKDLPAPIERTISKPK
GSVRAPQVYVLPPPEEEMTKKQVTLTCMVTDFMPEDIY
VEWTNNGKTELNYKNTEPVLDSDGSYFMYSKLRVEKK
NWVERNSYSCSVVHEGLHNHHTTKSFSR



LCDR1 LCDR2
DIVLTQSPSSLSASLGDTITITCHASQNINVWLSWYQQKPGNIPKLLIYKASNLHTGVPSRFSGSGS
GTGFTLTISSLQPEDIATYYCQQGQSYPLTFGGGTKLEIK
LCDR3

HCDR1 HCDR2
EVKLQESGGGLVQPGGSLKLSCATSGFTFSDYYMYWVRQTPEKRLEWVAYISNGGGSTYYPDTV
KGRFTISRDNAKNTLYLQMSRLKSEDTAMYYCARHGGYYAMDYWGQGTTVTVSS
HCDR3

A close-up of the variable heavy (teal) and light (orange) domains are shown above. Full heavy and light chain sequences are shown with the variable domain sequences in bold. The sequence motifs that will help identify N- and C-termini for each of these domains are underlined. CDRs are also depicted in both images above. There are several CDR definitions in the literature, and sources describing the different definitions and associated numbering conventions can be found in the **References** section. RFantibody was trained on CDRs following the Chothia definition. In this example, Chothia-defined light chain CDRs are in yellow and heavy chain CDRs in green.

A note about nanobodies

Nanobodies are the single variable domains derived from heavy chain-only antibodies naturally produced by camelids and cartilaginous fish. RFantibody is validated for the design of camelid-like nanobodies, or VHHs. These, like conventional antibody VHs, contain three CDRs. The improved solubility, ease of expression, and modularity of VHHs may make them a more appropriate format than a VH/VL pair, depending on the application. All of the instructions and tools in this tutorial apply to both conventional antibodies and VHHs.

Setup

Create a working directory to contain all your work on the tutorial files:

```
mkdir myfiles
cd myfiles
```

While RFantibody is normally installed with UV, on the workshop machines we've installed it as a conda environment.

```
conda activate rfantibody
```

Note on RFantibody installation

The RFantibody pipeline can be installed from the Github repository at https://github.com/RosettaCommons/RFantibody.

To properly use it, you need to patch some of the files. (If you're using the pre-installed version during the workshop, this has already been done for you.):

- `src/rfantibody/rfdiffusion/inference/model_runners.py` replace `base_complex_finetuned_BFF_9.pt` on line 75 with `RFdiffusion_Ab.pt`
- `scripts/proteinmpnn_interface_design.py` replace `/home/weights/ProteinMPNN_v48_noise_0.2.pt` on line 45 with the path to the installed weights
- `src/rfantibody/rf2/config/base.yaml` replace `/home/weights/RF2_ab.pt` on line 18 with the path to the installed weights.

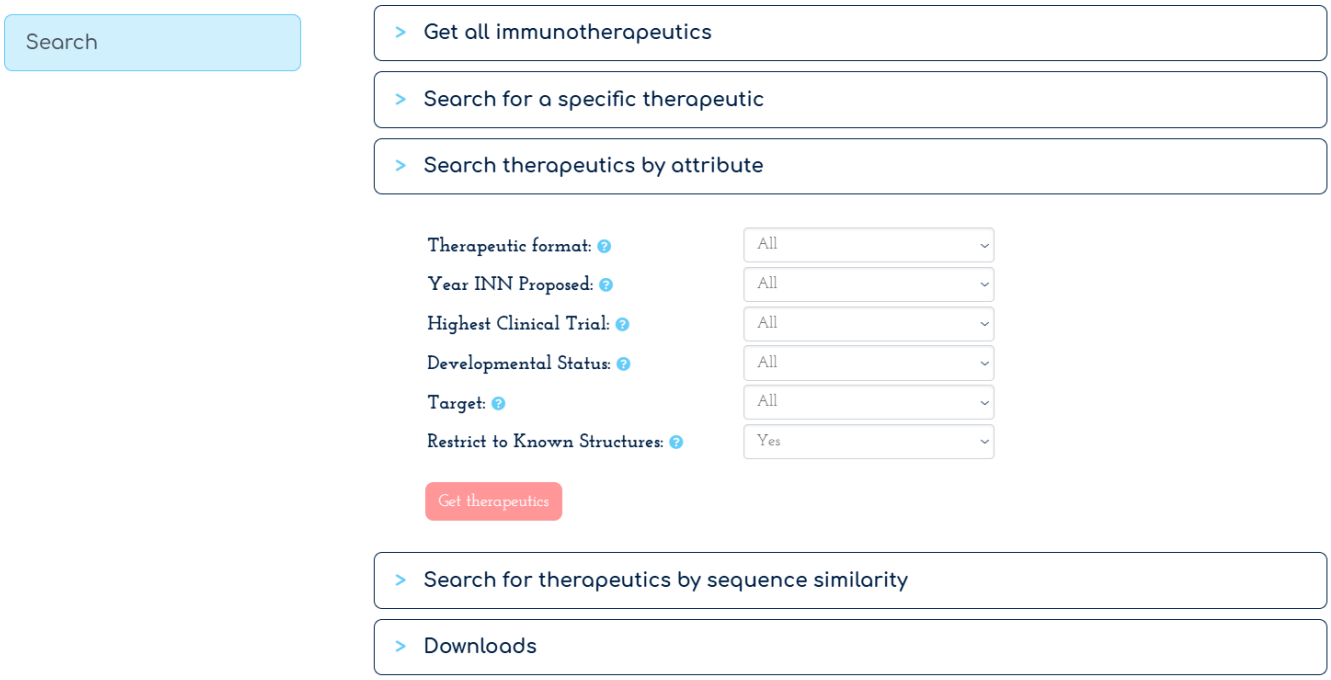
Input Preparation

There are two things we need: our **antibody framework** and our **target structure**.

Antibody framework

You will need to select your frameworks. It's a good idea to choose multiple antibody frameworks as starting points, since different frameworks can often support different types of CDR loops – allowing for more diversity allows for more shots on goal. We have the following cases, depending on whether you already have an antibody in mind:

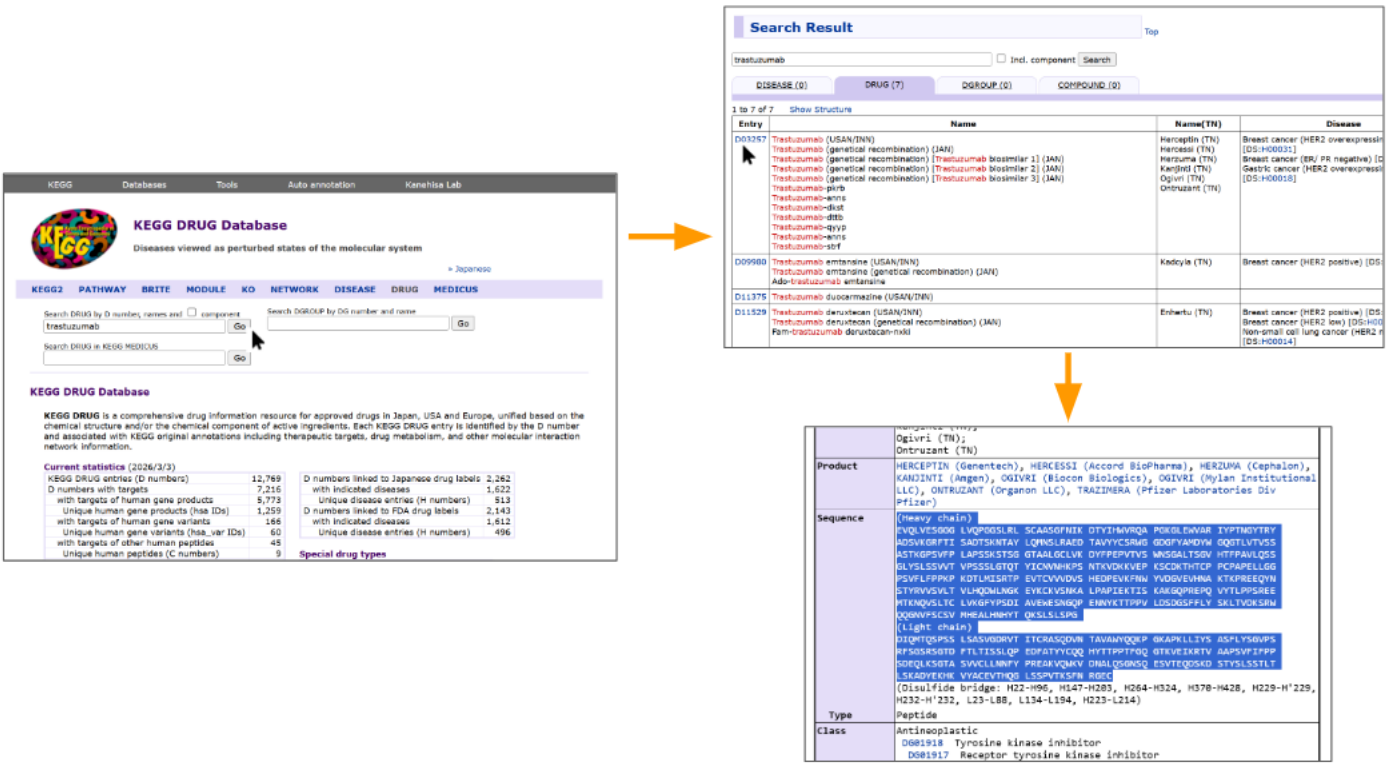
Case A: If you don't know what you want If you want to find some validated heavy-light chain pairs with known structures, Thera-SAbDab is a good place to start:



Make sure you download the **Chothia-numbered** PDB. The following is an arbitrary example:

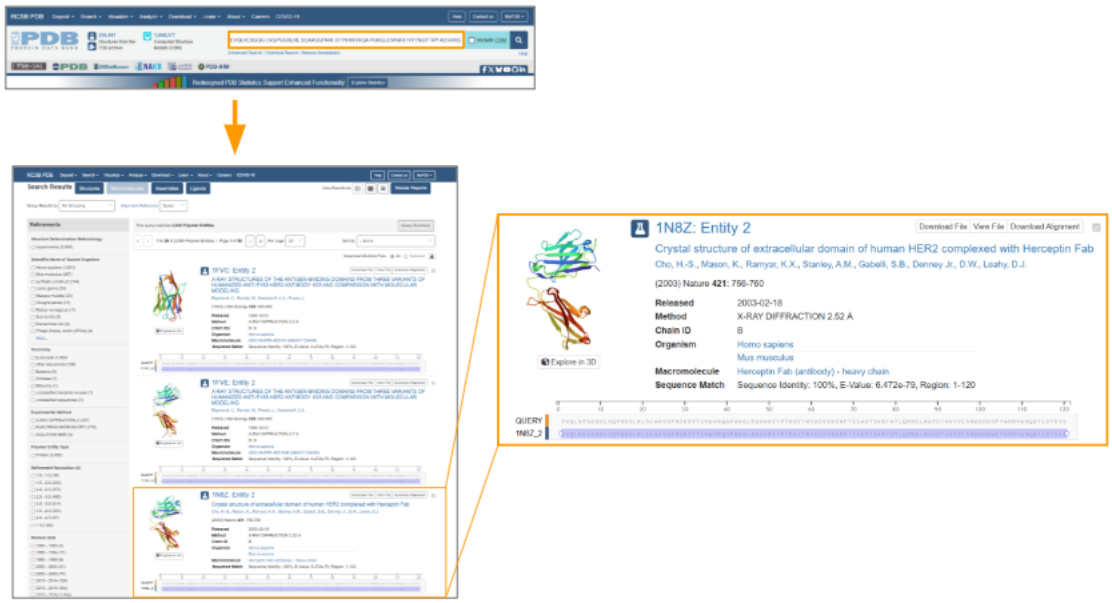
If you are starting with a full IgG or Fab, we strongly recommend truncating to the variable domains for quicker RFdiffusion and structure prediction runs. The heavy chain variable domain will likely begin with "EV" and end with "TVSS" and the light chain will usually begin with "DIQ" or "DIV" and end in "LEIK" or "VEIK" or something similar (see **Background**).

Case B: You know that you want a specific therapeutic antibody, but you don't know the structure Say you know that you want trastuzumab. You can find the exact sequence in the KEGG drug database:

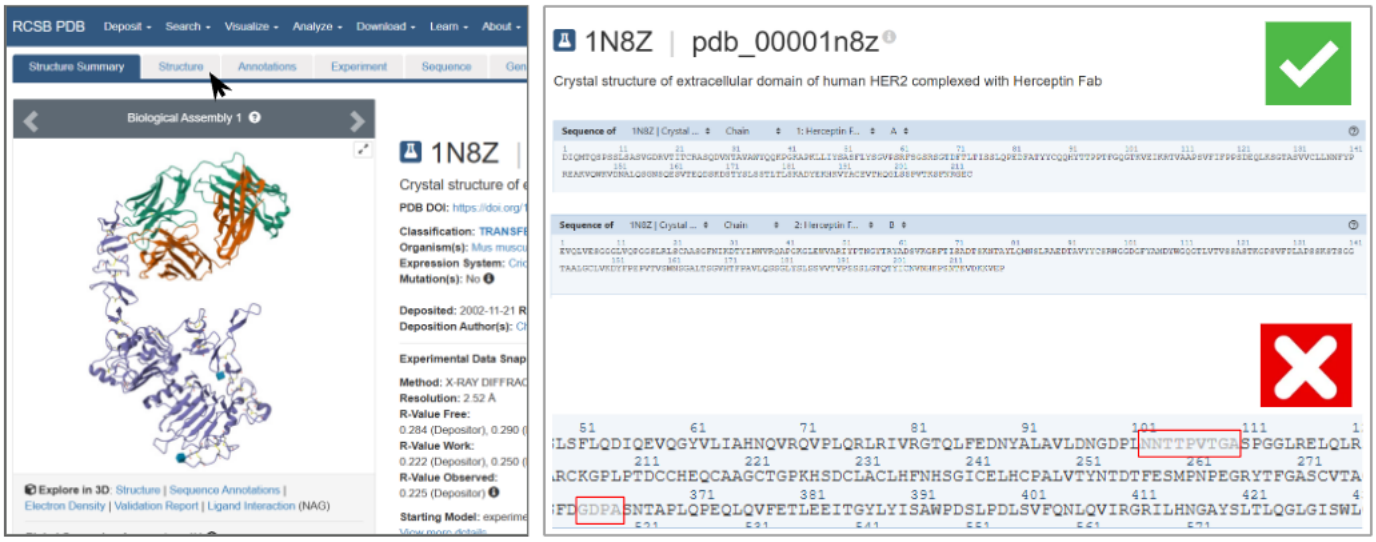


In KEGG, copy the variable domain sequences as described above and paste one chain at a time into the search bar on the

RCSB website. Look for a structure with 100% sequence identity. For this tutorial, we'll select 1N8Z:



Also check that all of the variable domain framework (non-CDR) residues are solved–to confirm this, you can look at the structure and check that none of the framework residues are greyed out. The following shows examples of what you want to see and want to avoid:



You should be able to find a structure that has 100% sequence identity to both chains. Note the PDB ID and search for it in SAbDab by going to **Search Structures > Search for a specific PDB entry**, then download the Chothia-numbered version of the PDB.

Preparing the tutorial target structure **Download the Chothia-numbered version of 1N8Z from SAbDab, as in Case B above.**

For computational efficiency, we recommend that you crop to just the VH + VL if you are designing an Fv, or crop to just VH if you are designing a nanobody. Either:

- Open the pdb file in VSCode, vim, etc and delete all regions not part of the VH+VL (or VHH)
- Open the pdb file in PyMOL, select only VH+VL (or only VHH), copy selection to new object, de-select original pdb in right sidebar, export current molecule as pdb.

For 1n8z, the VH domain is B1-B113 and the VL is A1-107

Save the trimmed structure as `1n8z_Fv.pdb`.

The antibody-finetuned version of RFdiffusion in RFantibody requires an HLT-remarked framework structure as input. This can be generated using the script provided with RFantibody

```
python ~/rosetta_workshop/RFantibody/scripts/util/chothia2HLT.py --heavy B --light A 1n8z_Fv.pdb
```

Target

Crop your target protein to just the region around the **epitope** you want to bind — this makes diffusion more compute-efficient. Do this cropping in PyMOL, similar to how you prepped the antibody framework.

For this tutorial, we'll be targeting the sialic acid binding site of influenza H7N9 hemagglutinin. This is chain E residues 47-260 from PDB id 6d8b. (Note that you may need to relabel the chain such that it isn't H/L.)

Download 6d8b from https://www.rcsb.org and trim it as above, saving it as 6D8B_trim.pdb

Running RFdiffusion

The first step in RFantibody is to generate antibody-target docks using an antibody-finetuned version of RFdiffusion.

RFantibody takes a list of desired CRF loop lengths. A typical approach is to keep a smaller length range for CDR H1 and H2, and leave more diversity — a wider range — for CDR H3, since CDR H3 tends to be the most important loop for binding specificity. For example, `"H1:7,H2:7-8,H3:5-17,L1:7,L2:7-8,L3:7-12"` indicates that you want to sample HCDR1 loops of exactly 7 aa, HCDR2 loops of 7 or 8 amino acids, HCDR3 lengths of 5-17 amino acids, etc.

You should also specify the "hotspot" residues. These are the residues which RFdiffusion will explicitly place in the epitope (the residues which the antibody should directly contact). These should be numbered according to your template input structure. (For this tutorial, we're using residues in the sialic acid binding pocket: E142, E174 and E217

```
mkdir -p rfdiff_out/
rfdiffusion -f 1n8z_Fv_HLT.pdb -t 6D8B_trim.pdb \
    -l "H1:7,H2:7-8,H3:5-17,L1:7,L2:7-8,L3:7-12" \
    -h "E142,E174,E217" -o rfdiff_out/1n8z_6d8b_ -n 1
```

The `-n 1` specified 1 output structures, which will be named with a `1n8z_6d8b_` prefix in the rfdiff_out/ directory

On the workshop machines, because of the lack of GPUs, this output structure should take ~1 hour. You can work ahead with the provided example output(s).

Once it finishes, you'll see your output files. At this point, you can take a look in PyMOL — just keep in mind that there will be **no sidechains** at this stage, since we haven't run ProteinMPNN yet. But you can look at the overall dock and the CDR loop conformations to get a sense of whether the designs look reasonable.

Running ProteinMPNN

Now that we have our backbones, we're going to use **ProteinMPNN** to design sequences for them. Essentially, what ProteinMPNN does is take a protein backbone — just the 3D coordinates, no amino acid identities — and predicts sequences that would fold into that structure.

A few tips on hyperparameters:

- It can help to **generate more sequences per backbone** — this gives you more candidates to evaluate downstream.
- You can **increase the temperature** to get more sequence diversity, but try to keep it at or below **0.3** — higher temperatures tend to lower design quality.
- **Omitting residues:** We strongly recommend omitting **cysteines** in antibody and VHH design to avoid potential sites of oxidation and inadvertent disulfide bonds. It's also a good idea to omit methionine from the CDRs, since methionine is also prone to oxidation.

```
cp -r ../outputs/rfdiff_out_example/ .  # Use the pre-provided inputs.

proteinmpnn --loops "H1,H2,H3,L1,L2,L3" --omit-aas CMX -n 2 \
    -i rfdiff_out_example/ -o protein_mpnn_out/
```

One more thing: While ProteinMPNN will redesign the sequence, it does not place sidechain atoms and does not alter the backbone. As such, the atom coordinates of the output at this stage will be identical to the RFdiffusion output. So hold off on evaluating your designs until after the next step.

Predicting the Structure of Your Designs

Now we're going to predict the structure of our designed sequences. While the original RFantibody paper used RFdiffusion's own scoring to filter designs, a retrospective analysis found that **AlphaFold3's ipTM metric** had better correlation with experimental binding for these antibody designs. So we recommend using AlphaFold3 for this step. For convenience, a version installed locally from https://github.com/google-deepmind/alphafold3 works best, but for a few designs using the public server at https://alphafoldserver.com/ is also possible.

That said, there are a few situations where you might not be able to use AF3:

1. You are restricted by the AlphaFold license. Use of AlphaFold3 (either the server or the local version) is governed by certain licensing terms which restrict not only who can use the service (e.g. restrictions on commercial use) but also restricts what you can do with the predicted structures. Your situation may fall under those restrictions.
2. Your target is **too large** (over 5,000 amino acids) and can't be reasonably cropped
3. Your target is a **viral target or otherwise restricted** by the AlphaFold server, and you don't have access to a local AF3 installation
4. You've **hit the prediction limit** — the AF3 server caps you at 30 predictions per day per email address

For any of those cases, we've included a section to run predictions with **RF3** instead. Note that prediction of Antibody/Antigen structures remains one of the harder tasks for structure prediction. As such, we recommend using one of the most recent generation ofr structure prediction programs, even if earlier versions suffice for your other use cases.

To run with the AF3 web server:

Use the script to prepare the input JSON files for the server. This takes the directory contianing your ProteinMPNN output PDBs, as well as the PDB with the target as you want to model it. (Versus your potentially highly trimmed epitope structure.)

```
../scripts/make_af_jsons.py protein_mpnn_out/ 6D8B_trim.pdb af_jsons/ --server
```

This will generate a series of JSON input files, which you can upload to https://alphafoldserver.com. 1. Run the function cells.
2. Copy the path to your ProteinMPNN output directory and paste it into the appropriate cell.
3. Paste the sequence of your target chain(s) into the dictionary in that same cell — give each chain a unique ID, and we recommend starting at "A". Skip "H" and "L" though, since we want to reserve those for the antibody chains. Add the path where you want the JSON file to be saved, and add a random number as a seed.
4. Run the cell, download the JSON file, then head over to the AlphaFold3 web server, upload the JSON, and submit your predictions.

To run AF3 from the command line:

The setup is effectively the same, but omit the `--server` from the `make_af_jsons.py` script, and pass those input JSONs to your local AF3 installation. We've linked the AF3 GitHub here if you need to reference the setup instructions.

Running with RF2

RFantibody comes with a version of RosettaFold2 specifically tuned for antibody predictions. The final step of the pipeline is to use the antibody-finetuned RF2 to predict the structure of the sequences we just designed. We then assess whether RF2 is confident that the sequence will bind as we designed.

NB: To get additional samples, vary the seed (`-s`) parameter.

```
rf2 -i protein_mpnn_out/ -o rf2_out/ -s 784194
```

Analyzing Your Results

Download your results from the AF3 server, or from wherever you ran RF3. You can visually inspect the predicted structures in PyMOL to get an initial sense of the dock and whether the CDR loops look reasonable.

Beyond the visual check, the two main metrics you want to look at are **ipTM** and **pAE**:

- **ipTM** (interface predicted TM-score) is a confidence metric for the **interface** between your antibody and target. Higher is better — values closer to 1 indicate AF3 is more confident that the two chains interact the way it's predicting.
- **pAE** (predicted aligned error) is a measure of **positional uncertainty**. Lower pAE values at the interface mean AF3 is more confident about the relative positions of the two chains.

You can also calculate the **RMSD** between your RFdiffusion backbone and the AF3 prediction. If the RMSD is low, that means two independent models — RFdiffusion and AlphaFold3 — agree on what the structure looks like, which gives us a lot more confidence that it's a real, stable structure.

Now go design some antibodies! If you run into any issues, please let us know. Good luck!

References

- Manuscript describing the RFantibody pipeline: https://www.nature.com/articles/s41586-025-09721-5
- ProteinMPNN manuscript: https://www.science.org/doi/full/10.1126/science.add2187
- AlphaFold3 manuscript: https://www.nature.com/articles/s41586-024-07487-w
- ANARCI, for antibody structure renumbering: https://academic.oup.com/bioinformatics/article/32/2/298/1743894
- Antibody numbering schemes and CDR definitions: http://www.bioinf.org.uk/abs/info.html
- Review on antibody numbering schemes (Zhu et al, 2025): https://academic.oup.com/peds/article/doi/10.1093/protein/gzaf005/8102903?guestAccessKey=#524740074