

RFdiffusion: Creating Binders & Identifying Protein-Protein Interactions

INTRODUCTION

For today's workshop, we're going to be running through an example of how to create binders using RFdiffusion, and do some pre-processing using custom scripts. We'll also show how this may be used to find structurally similar proteins that already exist. RFdiffusion is available in a variety of formats including a via command line, [Google Colab Notebook](#), and through NVIDIA (<https://build.nvidia.com/ipd/rfdiffusion>). In this session we'll get you using the command line implementation on Atlas, and the NVIDIA implementation. Unlike in the FoldSeek session, I don't recommend trying to use RFdiffusion without GPUs.

SET UP [## Estimated runtime: < 5 minutes]

1. Login to Atlas Open OnDemand using your web browser (<https://atlas-ood.hpc.msstate.edu/>) and navigate to your working directory for this workshop. Mine, for example, is in the shared directory under my username (Files > /90daydata > Change directory > /90daydata/shared/olivia.haley)
2. Once in your working directory, select **Open in Terminal**. A new window should open.
3. Copy the shared directory containing the scripts and structures for this demo to your working directory.

```
# Copy the shared directory from the shared folder
cp -r /90daydata/shared/protein_structure_workshop/RFdiffusion/ .
```

```
# Navigate to the copied directory and view its contents
cd RFdiffusion
ls -ltr
```

```
drwxr-xr-x 2 olivia.haley olivia.haley 4096 Nov  8 12:16 models
drwxr-xr-x 2 olivia.haley proj-maizegdb 4096 Nov 12 12:07 mock_binders
-rwxr-xr-x 1 olivia.haley olivia.haley 367821 Nov 12 12:45 7BNT.pdb
-rwxr-xr-x 1 olivia.haley olivia.haley 5324 Nov 12 14:36 rfdiffusion_env.yaml
drwxr-xr-x 3 olivia.haley olivia.haley 4096 Nov 12 18:27 example_outputs
-rwxr-xr-x 1 olivia.haley olivia.haley 80108 Nov 12 18:28 AF-Q7XJV3-F1-model_v4.pdb
drwxr-xr-x 2 olivia.haley olivia.haley 4096 Nov 12 18:29 rice_DB
drwxr-xr-x 2 olivia.haley olivia.haley 4096 Nov 13 13:51 workshop_scripts
```

4. Create the conda environment for this workshop. Note that the environment for RFdiffusion can be tricky to set up from scratch due to its dependencies. You also may experience a long running time when downloading the model weights. We've saved you time by pre-downloading the model weights into the *models* directory, and we'll set up the environment together.

```
# Create the conda environment and setup RFdiffusion
sbatch workshop_scripts/s1_setup.sh
```

TUTORIAL

In this tutorial, we're going to be working with a well-known fungal effector called Avr-PikD. It is a protein from the organism *Magnaporthe oryzae* which causes rice blast disease in rice. Its structure has already been determined and there is a good amount of evidence for its protein binding residues, making it a good candidate for high confidence predictions. Today we'll be making only 10 binders for this protein, but in reality you may want to make between hundreds to thousands of binders.

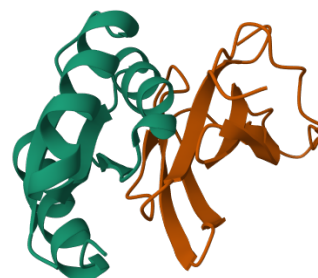
Step 1. [Optional] Create 10 binders for Avr-PikD

When used for binder structure prediction, RFdiffusion will output the binder complexed with the input protein (**bottom left**) in PDB format, as well as a .TRB file containing the metadata for the complex. **In practice, I would recommend using the a100s, this script with the mig7 partition may take more than an hour to run. So because of this, we're not going to run the diffusion step together.** Instead, we're going to use a directory of binders called *mock_binders* that I generated beforehand. Below is an example of one of the mock binder complexes.

```
#Edit the script as needed
sbatch workshop_scripts/s2_run_rfdiffusion.sh

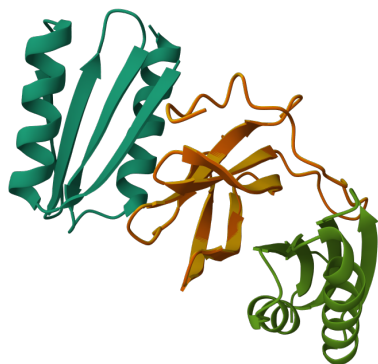
#View an example of the outputs
ls -l mock_binders/
```

```
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_10.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_1.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_2.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_3.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_4.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_5.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_6.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_7.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_8.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 42612 Nov 12 12:19 mock_binder_9.pdb
```



Step 2. Pre-filter the binders based on the anticipated protein contacts ('hotspots')

[## Estimated runtime: < 1 minute]



If we look at the structure of our binders in Mol*, we'll also see that RFdiffusion generated a variety of protein structures that appear to interact with different residues of our query protein (**example on the left**, orange/yellow=query protein). This illustrates the point that RFdiffusion will not use all of your input 'hotspots' when generating a binder. According to its GitHub, only 0-20% of these hotspots will be passed during binder generation.

We want to eliminate binders on the lower end of this threshold, and keep only binders where an interaction is most likely to occur. To do this, we'll use the distance between the carbon atoms at the query hotspots and target protein backbone.

Step 2 (continued)

This script filters out binders where there are little to no interactions occurring at the pre-defined hotspots. The .PDB files containing the 'acceptable' binders will be split into their chains, and only the binder chain (Chain A) will be placed into a directory called *accepted_binders*. The script also generates a .TSV file containing the residue distances.

```
#Run the script
sbatch workshop_scripts/s3_filter_and_extract_binders.sh
```

We can see that not all of the binders passed our initial, hotspot-based filtering.

```
#View the accepted binders
ls -l accepted_binders/
```

```
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_2_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_3_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_5_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_6_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_7_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_8_A.pdb
-rw-r----- 1 olivia.haley proj-maizegdb 0 Nov 12 16:44 mock_binder_9_A.pdb
```

And if we want, we can see the distances calculated for each binder and hotspot:

```
#Edit the script as needed
head residue_distances.tsv
```

design_name	residue	index	distance
mock_binder_6.pdb	HIS	46	8.347101
mock_binder_6.pdb	PRO	47	9.983383
mock_binder_6.pdb	GLY	48	7.5606647
mock_binder_6.pdb	ARG	64	10.824784
mock_binder_6.pdb	ASP	66	7.690056
mock_binder_6.pdb	ALA	67	8.581246
mock_binder_8.pdb	HIS	46	10.848546
mock_binder_8.pdb	PRO	47	12.08681
mock_binder_8.pdb	GLY	48	9.71202

Step 3. Run FoldSeek to identify homologous host protein substructures

You may be interested in which binders resemble experimentally determined protein-protein interactions. For this, we can use FoldSeek! There are too few binders in this example to get meaningful results, but it is something to keep in mind. In my experience, less than 100 binders will likely lead to spurious results. When interpreting the results, there are a few factors to keep in mind, such as:

- What type of protein is your anticipated PPI?
- Where is it located in the cell? Is the subcellular location similar to that of your query protein in the PPI?
- Does the target protein have annotations that could help you distinguish likely PPIs?

```
#Run FoldSeek against a target database of rice proteins and Filter the results by TM score
sbatch workshop_scripts/s4_foldseek_run_initial_query.sh
```

```
#Optionally view the results file
head AvrPikD_binder_target_foldseek_results.tsv
```


Step 4. RFdiffusion on the NVIDIA NIM microservice [## Estimated runtime: < 1 minute]

NVIDIA NIM is a part of NVIDIA, and provides a set of easy-to-use microservices designed to facilitate the use of generative AI tools. They also offer pre built containers and APIs to facilitate sharing models. For this part of the tutorial we're going to go through an example using NVIDIA <https://build.nvidia.com/ipd/rfdiffusion>. Compared to the Google Colab, this is the more simple implementation of RFdiffusion since you only need to specify your target protein, contigs, and hotspot residues. Depending on the user preferences, this can be disadvantageous as you cannot prepare more than one run at a time without a key.

If you choose to follow along, you'll need the following:

- Download the .PDB file (7BNT.pdb)
- Contig specs: C30-113/0 50-75
- Hotspot Residues specs: C46,C47,C48,C64,C66,C67

The screenshot displays the NVIDIA NIM interface for the 'ipd / rfdiffusion' microservice. The interface is dark-themed and includes a search bar at the top. The main content area is divided into 'Input' and 'Output' sections. The 'Input' section contains fields for 'Target Protein' (7BNT.pdb), 'Contigs' (C30-113/0 50-75), and 'Hotspot Residues' (C46,C47,C48,C64,C66,C67). A 'Diffusion Steps' slider is set to 15. The 'Output' section shows a 3D protein structure visualization. The interface also includes a 'Build with this NIM' button and a 'GOVERNING TERMS' section at the bottom.

Step 5A. Google Colab RFdiffusion: Setup [## Estimated runtime: < 5 minutes]

This is the most updated version of the Google Colab implementation of RFdiffusion (<https://colab.research.google.com/github/sokrypton/ColabDesign/blob/main/rf/examples/diffusion.ipynb>). You can log into Google Colab, or follow along here. The notebook is a good compromise between the simplicity of the NVIDIA implementation, and the more detailed command line code.

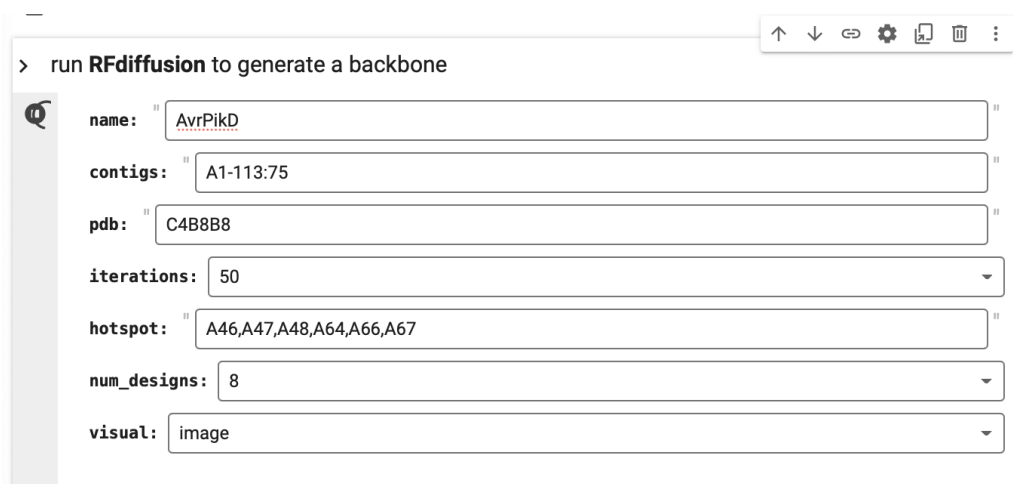
To get started, connect to a runtime. Every Google account has access to a limited number of free runtime units on the T4 GPUs. Run the block ' setup **RFdiffusion** ' (click the play button) to install the packages and dependencies. This should take around 3 minutes.

Step 5B. Google Colab RFdiffusion: Input Parameters for Binder Generation

[## Estimated runtime: < 7 minutes]

As it's running, you'll also see protein backbone being generated! After the entire run, you'll have the option of viewing each binder. If you'd like to follow along, the parameters you'll need for this step are:

name: AvrPikD	your name for the binders
contigs: A1-113:75	specifies the generation of binders 75 amino-acids long between residues 1-113 on the chain A of the input .PDB file.
pdb: C4B8B8	the name of the .PDB file to use. The Google Colab implementation of RFdiffusion is set up to automatically pull from the AlphaFold2 database if you use a UniprotKB identifier. So, In this instance, we're using the UniprotKB identifier for AvrPikD (C4B8B8)
hotspot: A46,A47,A48,A64,A66,A67	The chain and residues at the protein-protein interaction interface.
chains: A	The chain used for diffusion



The screenshot shows a code cell in a Google Colab notebook titled "run RFdiffusion to generate a backbone". The cell contains a series of input fields for the RFdiffusion command. The parameters are: name: "AvrPikD", contigs: "A1-113:75", pdb: "C4B8B8", iterations: "50", hotspot: "A46,A47,A48,A64,A66,A67", num_designs: "8", and visual: "image". The cell is currently in a "run" state, indicated by a play button icon on the left.

symmetry settings

symmetry: none

order: 1

chains: "A"

add_potential:

- symmetry='auto' enables automatic symmetry detection with [AnAnaS](#).
- chains="A,B" filter PDB input to these chains (may help auto-symm detector)
- add_potential to discourage clashes between chains

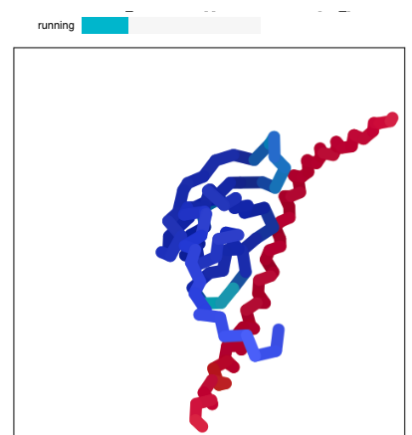
advanced settings

partial_T: auto


- specify number of noising steps (only used for the partial diffusion protocol)

use_beta_model:

- if you are seeing lots of helices, switch to the "beta" params for a better SSE balance.




> Display 3D structure

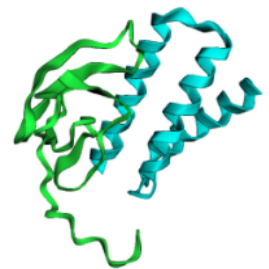
 animate: none

color: chain

dpi: 100

[Show code](#)

 design: 0



Step 5C. Google Colab RFdiffusion: Input Parameters for Binder Generation

[## Estimated runtime: < 7 minutes]

The Google Colab implementation takes binder generation one step forward by using AlphaFold2-multimer and ProteinMPNN. These pipelines design a sequence to the binder, and indicate the likelihood that the structure is 'real' based on AlphaFold2-multimer parameters.

num_seqs: 8	The number of sequences to sample for each binder
mpnn_sampling_temp: 0.1	The diversity of sequences to sample
rm_aa:	Amino acids to exclude from the sequences
use_solubleMPNN: False	Encourage solubility when designing protein sequences
intial_guess: True	The chain and residues at the protein-protein interaction interface.
num_recycles: 3	The number of recycles
use_multimier: True	The chain used for diffusion

run **ProteinMPNN** to generate a sequence and **AlphaFold** to validate

ProteinMPNN Settings

num_seqs: 8

mpnn_sampling_temp: 0.1

rm_aa: "C"

use_solubleMPNN:

- mpnn_sampling_temp - control diversity of sampled sequences. (higher = more diverse).
- rm_aa='C' - do not use [C]ysteines.
- use_solubleMPNN - use weights trained only on soluble proteins. See [preprint](#).

AlphaFold Settings

initial_guess:

- soft initialization with desired coordinates, see [paper](#).

num_recycles: 3

- for binder design, we recommend initial_guess=True num_recycles=3

use_multimer:

- use_multimer - use AlphaFold Multimer v3 params for prediction.

[Show code](#)

```
2024-11-01 21:59:13.646698: W external/xla/xla/service/gpu/nvptx_compiler.cc:893] The NVIDIA driver's CUDA version is 12.2 which is older than the PTX com
running proteinMPNN...
running AlphaFold...
design:0 n:0 mpnn:1.059 plddt:0.749 l_ptm:0.226 l_pae:17.654 rmsd:37.308 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:1 mpnn:1.181 plddt:0.668 l_ptm:0.105 l_pae:21.804 rmsd:37.916 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:2 mpnn:1.037 plddt:0.843 l_ptm:0.307 l_pae:15.196 rmsd:37.070 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:3 mpnn:1.182 plddt:0.580 l_ptm:0.097 l_pae:22.459 rmsd:35.770 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:4 mpnn:1.074 plddt:0.798 l_ptm:0.167 l_pae:19.379 rmsd:37.568 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:5 mpnn:1.053 plddt:0.751 l_ptm:0.138 l_pae:19.578 rmsd:37.453 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:6 mpnn:1.059 plddt:0.792 l_ptm:0.087 l_pae:22.830 rmsd:29.548 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:0 n:7 mpnn:1.041 plddt:0.585 l_ptm:0.195 l_pae:19.086 rmsd:37.560 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:1 n:0 mpnn:1.337 plddt:0.505 l_ptm:0.137 l_pae:22.948 rmsd:28.976 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/E
design:1 n:1 mpnn:1.400 plddt:0.416 l_ptm:0.118 l_pae:21.890 rmsd:30.630 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/E
design:1 n:2 mpnn:1.349 plddt:0.474 l_ptm:0.137 l_pae:21.557 rmsd:29.616 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/D
design:1 n:3 mpnn:1.355 plddt:0.489 l_ptm:0.149 l_pae:21.048 rmsd:31.144 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/P
design:1 n:4 mpnn:1.373 plddt:0.547 l_ptm:0.208 l_pae:21.393 rmsd:23.728 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/P
design:1 n:5 mpnn:1.376 plddt:0.465 l_ptm:0.129 l_pae:21.407 rmsd:28.987 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/P
design:1 n:6 mpnn:1.332 plddt:0.527 l_ptm:0.143 l_pae:21.024 rmsd:23.773 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/E
design:1 n:7 mpnn:1.333 plddt:0.481 l_ptm:0.144 l_pae:21.865 rmsd:28.970 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/P
design:2 n:0 mpnn:1.084 plddt:0.492 l_ptm:0.090 l_pae:23.100 rmsd:34.370 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:1 mpnn:1.068 plddt:0.642 l_ptm:0.091 l_pae:23.179 rmsd:20.780 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:2 mpnn:1.110 plddt:0.608 l_ptm:0.085 l_pae:23.763 rmsd:42.132 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:3 mpnn:1.093 plddt:0.640 l_ptm:0.080 l_pae:22.915 rmsd:20.972 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:4 mpnn:1.079 plddt:0.600 l_ptm:0.076 l_pae:23.622 rmsd:13.986 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:5 mpnn:1.144 plddt:0.780 l_ptm:0.091 l_pae:23.330 rmsd:37.175 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:6 mpnn:1.114 plddt:0.511 l_ptm:0.080 l_pae:23.745 rmsd:34.514 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/A
design:2 n:7 mpnn:1.027 plddt:0.609 l_ptm:0.078 l_pae:23.948 rmsd:22.542 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/G
design:3 n:0 mpnn:1.313 plddt:0.419 l_ptm:0.092 l_pae:23.360 rmsd:31.680 SGRNLYFGHMAAPARFCVYDGHLPATRVLLMYVRIGTTATITARGHEFEVEAKDQCKVILTKGQAPDWLAAEPY/I
```

Then you have the option to view and download the results. The download may take a while depending on the complexity and number of binders you create.

Display best result

Show code

design: best



