**PROTOCOL TO GENERATE PROTEIN BINDERS**

Most binder designs fail because of one of the following reasons:

* **Type I Error**: The structure **fails to fold** into the proper monomeric state.
* **Type II Error**: The structure folds into the proper monomer but **fails to bind** the target properly.

A diagram of a structure

AI-generated content may be incorrect.

Figure 1. Source: N. R. Bennett et al. Nature Communications (2023).

The following protocol will lead you through a successful binder design process, while minimizing poor designs. The protocol described herein aims to create **monomers**, whereas multimers are still not possible to generate with these tools. It is important to note **several caveats** of the protocol described below (i.e., RFdiffusion + ProteinMPNN + AF2 ‘initial guess’):

* Binders are biased towards **alpha helixes**, whereas beta strands are seldomly used.
* Binders cannot bind polar surfaces, **hydrophobic interactions** are necessary in most cases.
* Thousands of designs are needed to get enough valid designs for the experimental testing, which is computationally costly.

Besides this protocol, new binder design pipelines are being developed such as **BindCraft**:

* <https://github.com/martinpacesa/BindCraft>
* <https://www.youtube.com/watch?v=u5yijcBsonw>
* <https://www.biorxiv.org/content/10.1101/2024.09.30.615802v2>
* <https://levitate.bio/rosetta/bindcraft/ai/2025/02/20/bindcraft.html>

# 1. PDB download and cleaning:

1. Download the target PDB file.
2. Remove all the water molecules, ions, ligands, and any other molecules that are not our target protein. Save the cleaned PDB file, which will be used as input for backbone generation.

# 2. Backbone design:

Main Source: <https://github.com/RosettaCommons/RFdiffusion>

## 2.1. Theory:

Currently, the binder backbone is created using RFdiffusion (diffusion model). Other models are being developed to design backbones, so keep updated about new methods.

Binders can be designed in an **(i) unconditional** or (ii) **conditional** way. The former will create binders to any part of the protein that the model believes is targetable. However, whenever possible, it is **recommended to use the conditional binder design** because of its **higher success rate:**

A graph of different colored bars

AI-generated content may be incorrect.

Figure 2. Binder design in vitro success rate was up to 33% (IL-7Ra as a target) when using hotspot-conditioning. Source: J. L. Watson et al. Nature (2023).

A table with numbers and text

AI-generated content may be incorrect.

Figure 3. Binder design in silico success rate was higher in 4 out of 5 targets when using fold-conditioning. Source: J. L. Watson et al. Nature (2023), Extended Data Fig. 8.

Backbone generation can be conditioned using **(i) hotspot information** and/or (ii) **fold-conditioning information** (i.e., secondary structure and block-adjacency). These conditions will guide the denoising process to generate tailored binders targeting the desired region of a protein (i.e., hotspot) and with a specific folding structure (e.g., ferredoxin fold).

Several types of structures

AI-generated content may be incorrect.

Figure 4. A) Two hotspot sites in target proteins. C) Conditional folding. Source: J. L. Watson et al. Nature (2023), Extended Data Fig. 8.

Another reason to use the **conditional** design is because often, target proteins need to be cropped beforehand to reduce computational cost. Providing the whole of your target, uncropped, is going to make diffusion very slow if your target is big (and most targets-of-interest, such as cell-surface receptors tend to be *very* big). One tried-and-true method to speed up binder design is to crop the target protein around the desired interface location. BUT! This creates a problem: if you crop your target and potentially expose hydrophobic core residues which were buried before the crop, how can you guarantee the binder will go to the intended interface site on the surface of the target, and not target the tantalizing hydrophobic patch you have just artificially created?

We solve this issue by providing the model with what we call **hotspot** residues. The model readily learns that it should be making an interface which involves these **hotspot** residues. At inference time then, we can provide our own hotspot residues to define a **region which the binder must contact**. These are specified like this: *'ppi.hotspot\_res=[A30,A33,A34]'*, where *A* is the chain ID in the input PDB file of the hotspot residue and the number is the residue index in the input PDB file.

For example, in the article ‘J. L. Watson et al. Nature (2023)’ the authors used 3 hotspots to design a binder against PD-L1 (A56, A115, A123) which are specified in the supplementary material (page 97).

## 2.2 Practical Considerations for Binder Design:

RFdiffusion is an extremely powerful binder design tool but it is not magic. In this section we will walk through some common pitfalls in RFdiffusion binder design and offer advice on how to get the most out of this method.

**Picking hotspots:**

We normally **recommend** **between** **3-6 hotspots**, you should run a few pilot runs before generating thousands of designs to make sure the number of hotspots you are providing will give results you like. Importantly, the region in which those hotspots are located should be carefully selected.

Not every region on a target protein is a good candidate for binder design. For a site to be an attractive candidate for binding it should have **>~3 hydrophobic residues** for the binder to interact with. Binding to charged polar sites is still quite hard. Binding to sites with **glycans** close to them is also hard since they often become ordered upon binding and you will take an energetic hit for that. Historically, binder design has also **avoided unstructured loops**, it is not clear if this is still a requirement as RFdiffusion has been used to bind unstructured peptides which share a lot in common with unstructured loops.

In summary, your **3-6 hotspots** should be located in a region with **1)** **>~3 hydrophobic residues** nearby that will be part of the interface, and should avoid **2)** nearby **glycans** and **3) unstructured loops**.

**Truncating your Target Protein:**

RFdiffusion scales in runtime as O(N^2) where N is the number of residues in your system. As such, it is a very good idea to truncate large targets so that your computations are not unnecessarily expensive. RFdiffusion and all downstream steps (including AF2) are designed to allow for a truncated target. Truncating a target is an art. For some targets, such as multidomain extracellular membranes, a natural truncation point is where **two domains are joined by a flexible linker**. For other proteins, such as virus spike proteins, this truncation point is less obvious. Generally, you want to **preserve secondary structure** and introduce **as few chain breaks as possible**. You should also try to leave **~10A of target protein on each side of your intended target site**. We recommend using **PyMol** to truncate your target protein.

**Binder Design Scale:**

Usually, generating **~1,000 to ~2,000** backbone designs should be sufficient. However, some targets may require ~10,000 backbone designs.

The size of the binder can vary but a reasonable size is **100 amino acids**. For example, S. Vázquez Torres et al. Nature (2025) created binders of 100 residues. However, RFdiffusion works well creating backbones **up to 300 amino acids** (see Figure 5).

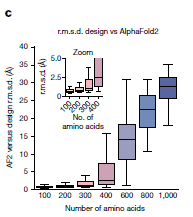


Figure 5. RMSD of binders designed by RFdiffusion vs AF2 prediction. J. L. Watson et al. Nature (2023).

**Noise:**

Reducing the noise added at inference improves designs, as shown in Figure 3. This comes at the expense of diversity, but, given that the scaffold sets are huge, this probably doesn't matter too much. We therefore recommend lowering the noise. **0.0** or **0.5** is probably a good compromise (see Figure 3). This just scales the amount of noise we add to the translations (noise\_scale\_ca) and rotations (noise\_scale\_frame) by, in this case, 0.5.

**Fold Conditioning:**

Something that works really well is **conditioning binder design** (or monomer generation) on particular topologies. This is achieved by providing (partial) **secondary structure** and **block adjacency information**. This permits 'low resolution' specification of output topology (i.e., I want a TIM barrel but I don't care precisely where residues are):

**A diagram of a structure

AI-generated content may be incorrect.**

Figure 6. We provide the following features to the model: (1) an [L,4] one-hot tensor where each position is assigned to a secondary structure type {helix, sheet, loop, mask} and (2) an [L,L,3] one-hot tensor (called the block adjacency matrix) where entries indicate membership in blocks that are within a distance cutoff of one another {non-adjacent, adjacent, mask}. Source: J. L. Watson et al. Nature (2023), Extended Data Fig. 4.

However, fold-conditioning is still difficult to apply, I have not found a clear way to create the secondary structure tensor and the adjacent matrix. Some information is shown in the GitHub of [RFdiffusion](https://github.com/RosettaCommons/RFdiffusion?tab=readme-ov-file#getting-started--installation), but its not very clear. Also, it seems that they are about to publish an article showing that fold-conditioning can be applied to guide the formation of b-strand pairing between the binder and the target. They claim that this allows binding polar regions of a target:

<https://www.biorxiv.org/content/10.1101/2024.10.11.617496v2.full>

**Output files of RFdiffusion:**

There are several output files:

1. **The .pdb file**. This is the **final prediction** out of the model. Note that every designed residue is output as a glycine (as we only designed the backbone), and no sidechains are output. This is because, even though RFdiffusion conditions on sidechains in an input motif, there is no loss applied to these predictions, so they can't strictly be trusted.
2. The **.trb file**. This contains useful **metadata** associated with that specific run, including the specific contig used (if length ranges were sampled), as well as the full config used by RFdiffusion. There are also a few other convenient items in this file:
   * details about mapping (i.e. how residues in the input map to residues in the output)
     + con\_ref\_pdb\_idx/con\_hal\_pdb\_idx - These are two arrays including the input pdb indices (in con\_ref\_pdb\_idx), and where they are in the output pdb (in con\_hal\_pdb\_idx). This only contains the chains where inpainting took place (i.e. not any fixed receptor/target chains)
     + con\_ref\_idx0/con\_hal\_idx0 - These are the same as above, but 0 indexed, and without chain information. This is useful for splicing coordinates out (to assess alignment etc).
     + inpaint\_seq - This details any residues that were masked during inference.
3. **Trajectory files**. By default, we output the full trajectories into the /traj/ folder. These files can be opened in pymol, as multi-step pdbs. Note that these are ordered in reverse, so the first pdb is technically the last (t=1) prediction made by RFdiffusion during inference. We include both the pX0 predictions (what the model predicted at each timestep) and the Xt-1 trajectories (what went into the model at each timestep).

## 2.3 Code

RFdiffusion is run through the “**run\_inference.py**” source code. It will use different trained models depending on the task (e.g., binder design, scaffold design, etc.). Model selection is handled automatically, no need to worry about this. You only need to think about the parameters to send to the model.

One of the most important parameters to introduce in the code is ‘**contigmap.contigs**’. This parameter tells the model which residues correspond to the target and which should be the size (exact or a range) of the binder. We also use this parameter to diffuse scaffolds. This is how to use ‘**contigmap.contigs**’:

* Anything **prefixed by a letter** corresponds to the **target’s chain letter** and **residues** in the input PDB file (e.g. A10-25 pertains to residues 'A10’ to 'A25’ of the chain ‘A’). Note that these **prefixed** residues **will not be diffused** because they correspond to the target.
* Anything **not prefixed** by a letter indicates **protein to be built** (e.g., binder or a scaffold). This can be input as an exact length (i.e., 70-70) or as a length range (i.e., 70-100). These length ranges are randomly sampled each iteration of RFdiffusion inference.
* To specify chain breaks, we use ‘**/0**’.

For example, if we want a 80 to 100 residue long binder for a target that has two chains (A with 98 residues and B with 56 residues) we would use the following ‘'contigmap.contigs’ = [A1-98/0 B1-56/0 80-100].

**Other** required **parameters** are:

* **Input folder**: contains the PDB file of the target protein.
* **Output folder**: where the model output will be saved.
* **Hotspots**: list of the selected residues that are hydrophobic or nearby of hydrophobic residues. The hotspots do not need to be hydrophobic residues themselves.
* **Noise:** Noise level that is added to the inference, the lower the better (Figure 3). 0 is default.
* **Number of backbone designs** to generate: 1,000 or more is a common value.

# 3. Sequence design:

Sequence design is done using the **ProteinMPNN-FastRelax** protocol. The code for this protocol can be found in: <https://github.com/nrbennet/dl_binder_design>

ProteinMPNN is provided the **complex structure** (i.e., **target + binder**) with the binder sequence masked and asked to assign the binder a sequence. The new sequence is then threaded back onto the binder structure and the complex structure is relaxed using Rosetta FastRelax. The relaxed complex structure can then be used as the input to ProteinMPNN to continue the cycle. (**Source**: N. R. Bennett et al. Nature Communications (2023))

The required **parameters** are:

* **Input folder**: The PDB files with the backbone designed by RFdiffusion. These PDBs should:
  1. Include the **target protein**, which is important because the residues in the interface of the binder protein should be more hydrophobic.
  2. The backbone of the designed binder should be the **first chain** of the PDB file (you can check this in PyMol).
* **Output folder**: The output PDBs do not have relaxed sidechains so the structures will look strange if you visualize them in PyMol, this is perfectly normal, the structures will look better after run though AF2.
* **Number of Relax cycles**: default is 1 cycle.
* **Sequences per backbone**: default is 1 cycle.
* **Sampling temperature**: default is 0.001.

The values of the **parameters** above are **based on best practices:**

* N. R. Bennett et al. Nature Communications (2023).
* J. L. Watson et al. Nature (2023)\_SuppMat.

# 4. Structure prediction:

The structure of the sequence designed by ProteinMPNN is predicted using **AlphaFold2 (AF2) ‘initial guess’ single-seq** protocol. The code for this protocol can be found in: <https://github.com/nrbennet/dl_binder_design>

AF2 ‘initial guess’ is a modification of AF2 that uses the ProteinMPNN sequence and the RFDiffusion backbone to predict a bound complex model. It essentially works by providing the target design to AlphaFold in the prediction step and biasing it towards the target. This significantly improves the success rate of the multimer prediction and these predictions are in turn highly valuable for filtering away designs which have Type II errors.

The required **parameters** are:

* **Input folder**: The PDB files with the designed sequences by ProteinMPNN. These PDBs should:
  1. Include the **target protein**.
  2. The designed binder with the sequence from ProteinMPNN should be the **first chain** of the PDB file (you can check this in PyMol).
* **Output folder**: The output PDBs have the structure (binder+target) predicted by AF2 as well as relevant data corresponding to **'plddt\_binder'**, **‘pae\_interaction’**, and **‘binder\_aligned\_rmsd’** which will be used in the filtering phase of the protocol.

Note: we can work with silent files instead of PDB files. A silent file can compress multiple PDB files which saves storage capacity and accelerates data processing.

# 5. Rosetta Delta Delta G (ddg) estimation (OPTIONAL):

**Source**: <https://github.com/ELELAB/RosettaDDGPrediction>

This step is not described in the articles outlining the protocol for binder design. Thus, it is an **OPTIONAL** step. However, some articles have applied it as an extra filter that defines in silico success: S. Vázquez Torres et al. Nature (2025).

Rosetta **Delta Delta G (ddg)** is an estimate of the binding energy of a complex (i.e., binder-target). It is computed by taking the difference of the energy of the complex and of the separated components. The lower the **ΔΔG**, the more stable and favorable will be the complex formation of the target and the binder.

**Note:** I’m not sure whether this part of the code works well, so be cautious if using it.

# 6. Binder design filtering:

**Sources**: S. Vázquez Torres et al. Nature (2025), N. R. Bennett et al. Nature Communications (2023), J. L. Watson et al. Nature (2023).

The generated binders are filtered on the basis of:

1. AF2-predicted aligned error (**PAE**) of interaction (**pae\_interaction**) **< 10**.
2. AF2-predicted local distance difference test (**pLDDT**) of the binder **> 80**.
3. AF2-predicted **rmsd <1Å** between the designed binder and the AF2 prediction.
4. **Optional**: Rosetta-estimated **Delta Delta G (ddg) < −40**.

Note: we only need the pLDDT and rmsd of the binder (i.e., monomer).

What you want is to get **enough designs** that **pass these filters** to fill a DNA order with these successful designs.

Predicted aligned error ([**PAE**](https://www.ebi.ac.uk/training/online/courses/alphafold/inputs-and-outputs/evaluating-alphafolds-predicted-structures-using-confidence-scores/pae-a-measure-of-global-confidence-in-alphafold-predictions/)) is a measure of how confident AF2 is in the relative position of two residues within the predicted structure. The **pAE of interaction between binder and target < 10** is a good **predictor** of a **binder working experimentally**, whereas designs that do not pass **pae\_interaction < 10** are not worth ordering since they will likely not work experimentally.

Aligning the backbone from RFdifussion to the AF2-predicted structure will give the **rmsd** value.

The predicted local distance difference test (**[pLDDT](https://www.ebi.ac.uk/training/online/courses/alphafold/inputs-and-outputs/evaluating-alphafolds-predicted-structures-using-confidence-scores/plddt-understanding-local-confidence/)**) is a per-residue measure of local confidence. It is scaled from 0 to 100, with higher scores indicating higher confidence and usually a more accurate prediction. On this basis, a pLDDT above 90 would be taken as the highest accuracy category, in which both the backbone and side chains are typically predicted with high accuracy. In contrast, a pLDDT above 70 usually corresponds to a correct backbone prediction with misplacement of some side chains.

# 7. Partial diffusion to optimize binders:

Sources: S. Vázquez Torres et al. Nature (2025) and <https://github.com/RosettaCommons/RFdiffusion>

After testing the binder candidates in vitro, we can **further optimize** their **binding affinity** using **partial diffusion**. For instance, S. Vázquez Torres et al. Nature (2025) showed how RFdiffusion can be extended for **binder refinement** by partial noising and denoising, which enables tuneable sampling around a given input structure. For peptide binder design, this enabled **increases in affinity** of nearly **three orders of magnitude** without high-throughput screening.

The AF2 models of the highest-affinity designs for each in vitro-tested binder are used as the inputs for partial diffusion. The designs are subjected to 10 and 20 noising time steps out of a total of 50 time steps in the noising schedule and subsequently denoised (“diffuser.partial\_T” input values of 10 and 20). Approximately **2,000** partially diffused designs should be generated for each target. The resulting library of backbones are then sequence designed using ProteinMPNN and Rosetta FastRelax, followed by AF2 ‘initial guess’. The resulting libraries are filtered again on the basis of AF2 PAE < 10, pLDDT > 80, rmsd <1Å and, optionally, Rosetta ddg < −40.