

A high-performance and modular protein design pipeline

Josh Hardy

Lucet Laboratory

New Medicines and Diagnostics Division

Walter and Eliza Hall Institute of Medical Research



Socials/Links

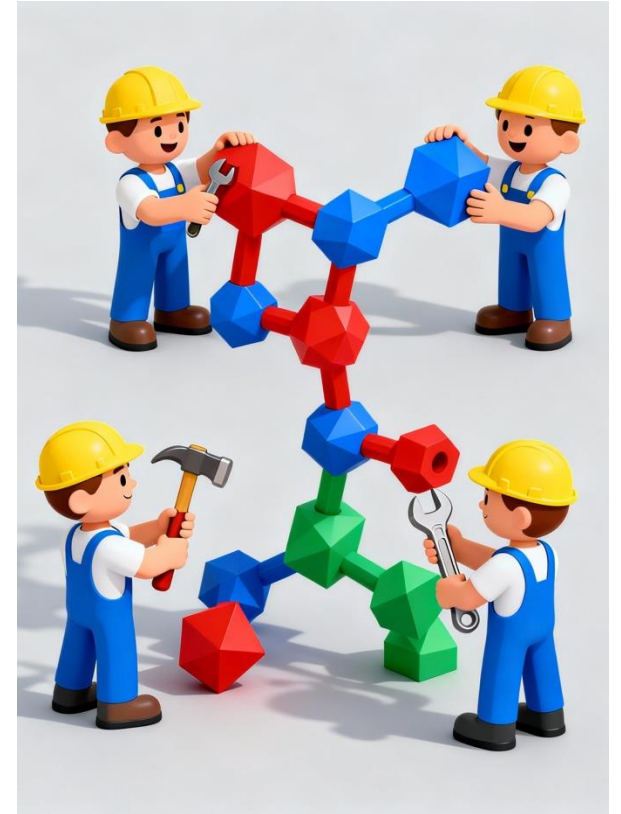
Presentation overview

1. Introduction to protein design
2. What ProteinDJ can do for you
3. Getting started in binder design
4. Behind the scenes: Development of ProteinDJ
5. How to install and run ProteinDJ

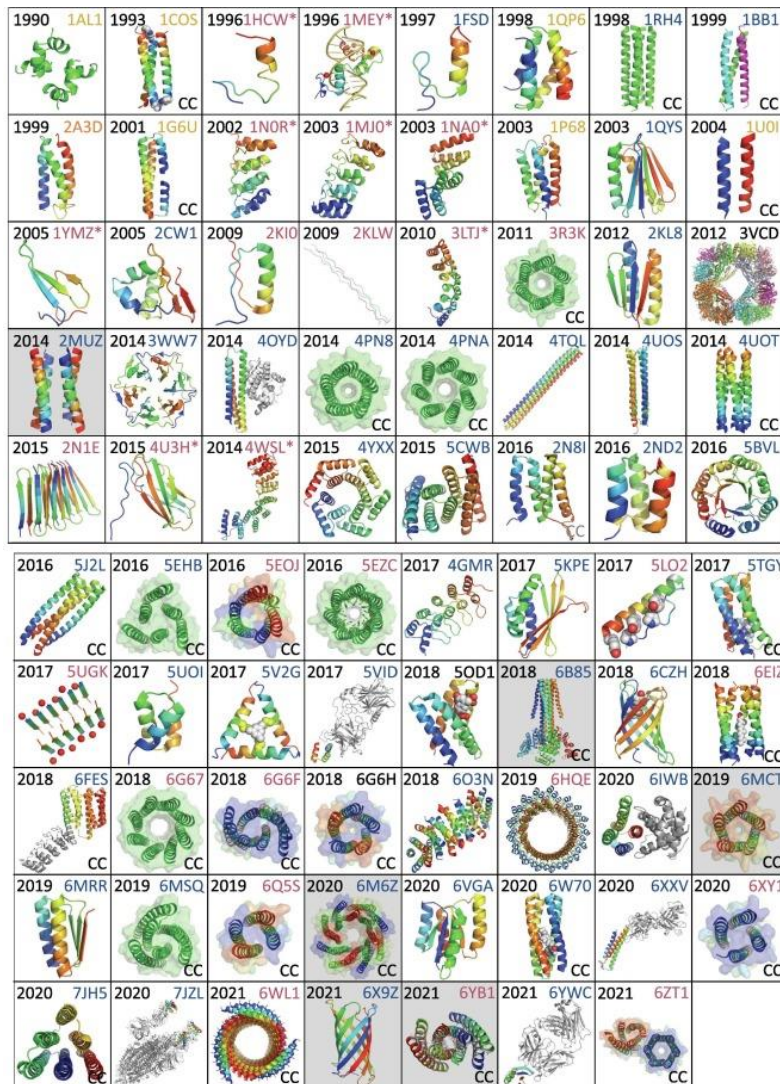
Introduction to Protein Design

Protein design comes in many flavours

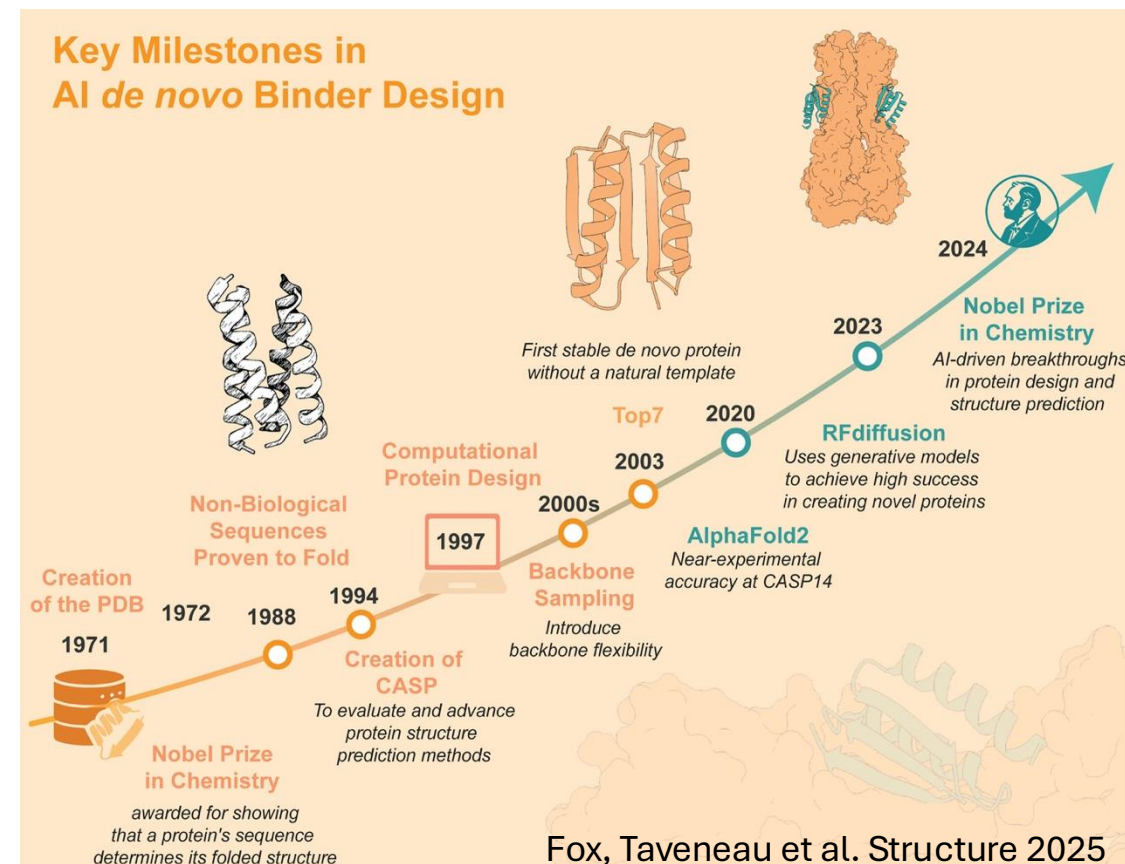
- We have been designing and modifying proteins for decades:
 - Protein fusions – concatenating sequences e.g. expression tags, solubility tags, domains
 - Mutations/deletions/insertions – to modify function
 - Grafting of binding loops in antibodies
- Many of these sequence changes will also result in a structural change and can affect protein stability and folding
- Designing new or ‘*de novo*’ proteins is the ultimate test of our understanding of the protein structure-sequence relationship
 - “What I cannot create, I do not understand” – Richard Feynman



The rise of *de novo* proteins



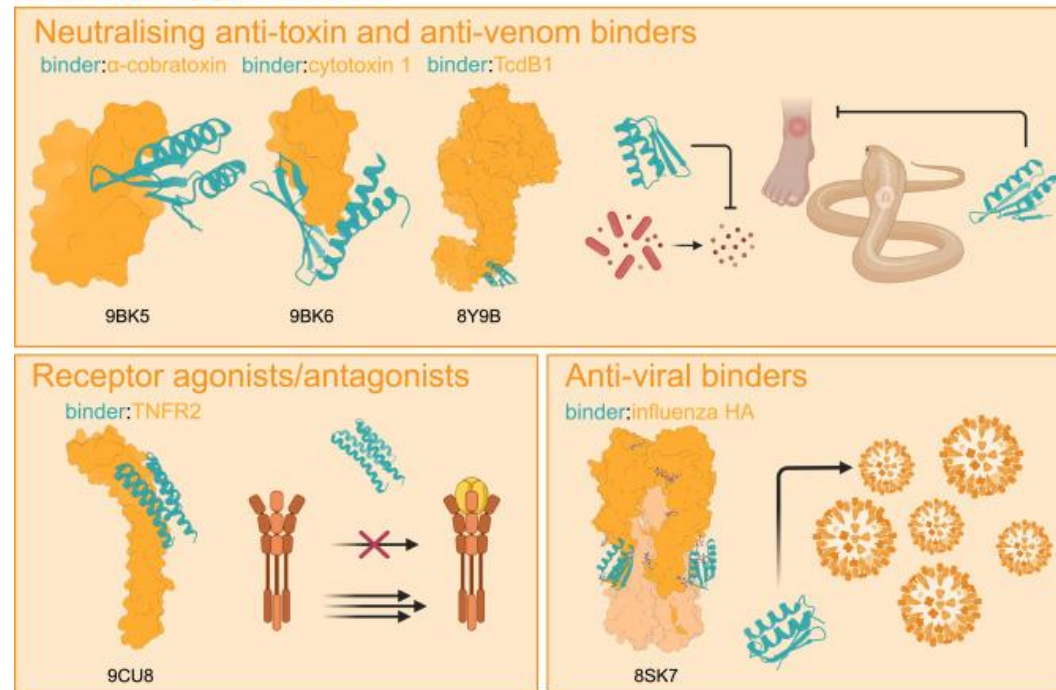
Deep learning methods have recently accelerated *de novo* protein design



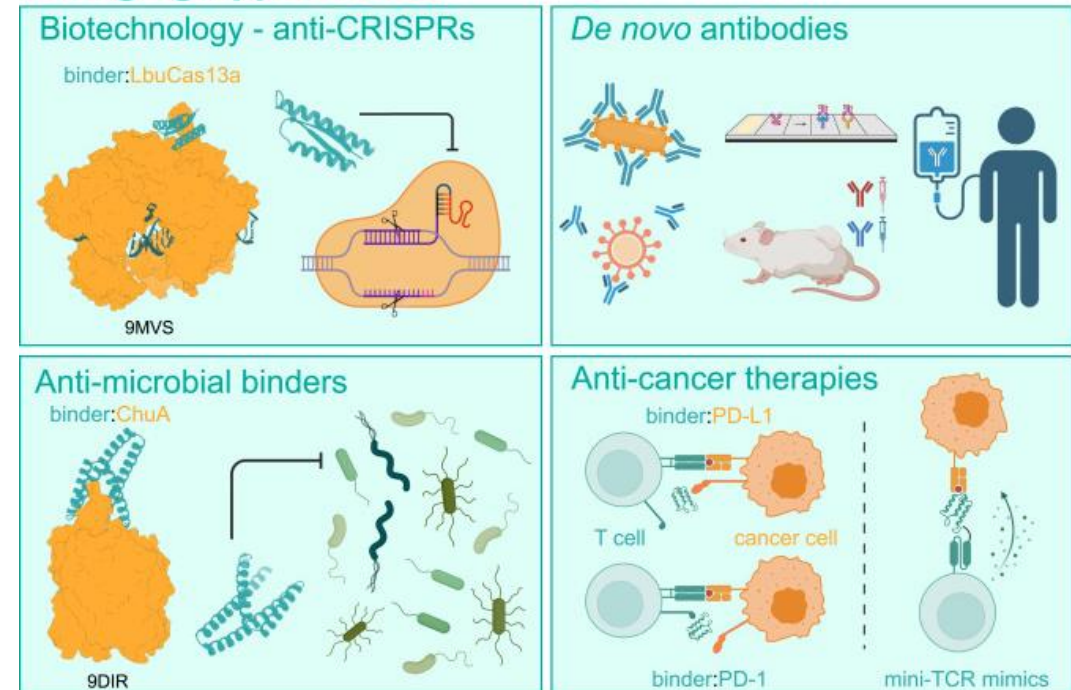
The untapped potential of *de novo* protein binders

- A subgenre of protein design is the creation of *de novo* protein binders that bind target proteins
- *De novo* protein binders have been used to solve problems in biology, chemistry and medicine

Current Applications



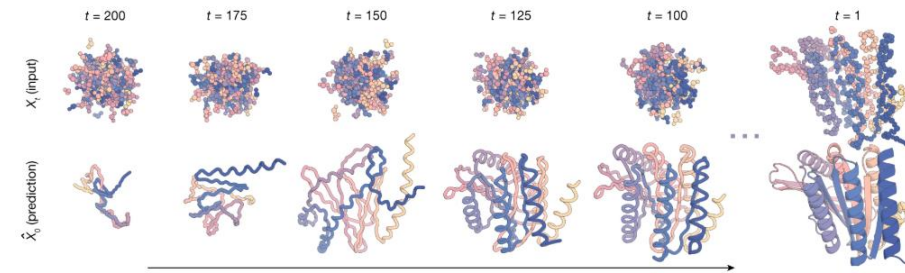
Emerging Applications



Software for *de novo* binder design

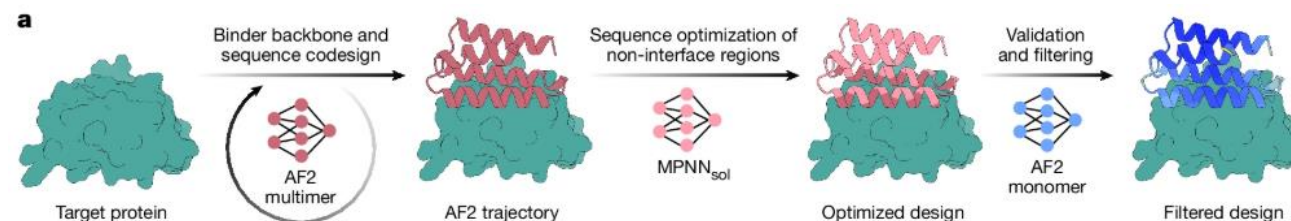
- A rapidly expanding list with different approaches and availability
- Generative methods – trained on PDB database to generate new folds, typically using diffusion from random noise

- **RFdiffusion**
- Chai-2
- PXDesign-d
- AlphaProteo



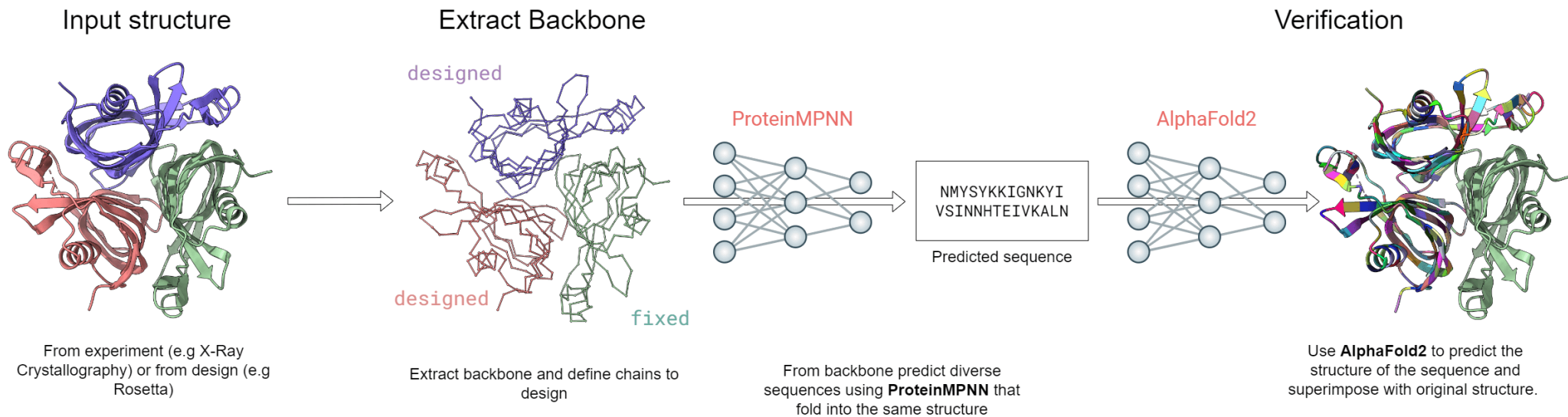
- Hallucination methods – iterative optimisation of sequences using structure prediction methods

- **BindCraft**
- BoltzDesign1
- PXDesign-h



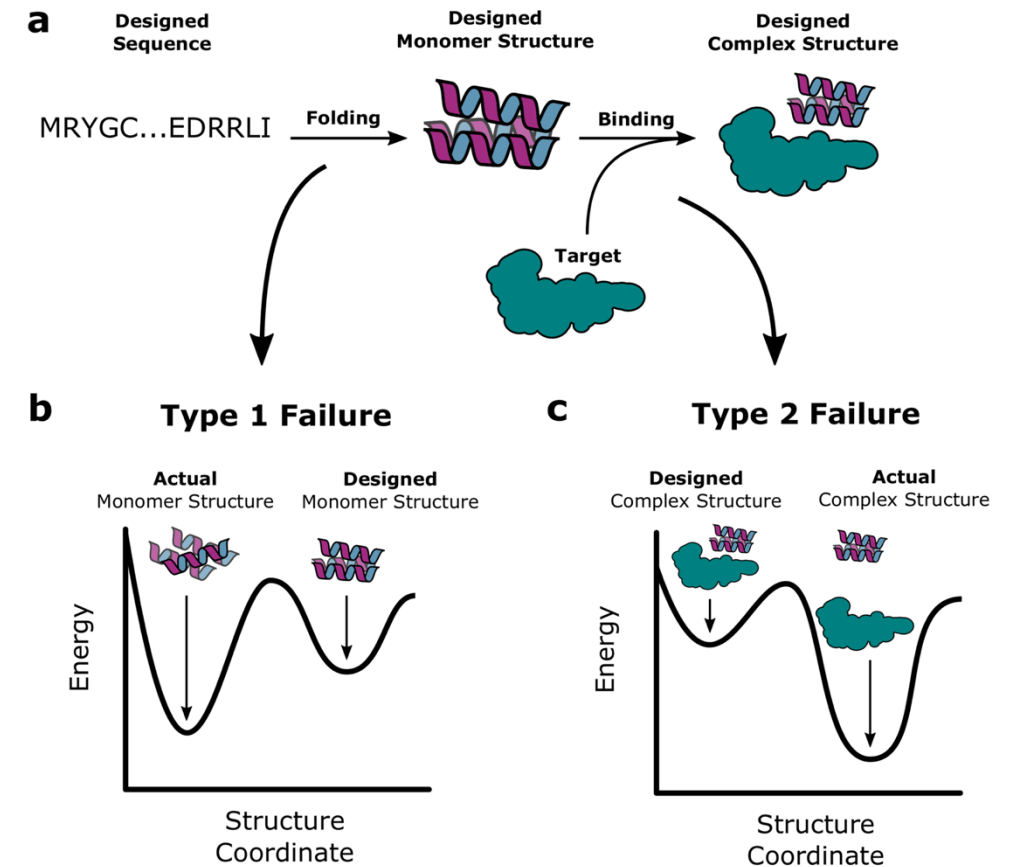
ProteinMPNN and structure prediction

- Some protein design tools (e.g. RFdiffusion) only generate a fold without a sequence or side-chains
- ProteinMPNN can design a sequence to match the fold. There are multiple variants e.g. SolubleMPNN, Ligand MPNN, Full-Atom MPNN
- We can test the designed sequence using structure prediction
 - i.e. Does the protein fold the same as the original design?



Challenge 1: Binder design can have a high failure rate

- Early *de novo* design campaigns required expression of hundreds or thousands of designs
- Structure prediction methods have improved the chance of success but we still need dozens of designs to test in the lab
- Success rates for structure prediction can vary dramatically, often 0-5%
- This requires the structure prediction of thousands of designs
 - Need for high-performance computing



Challenge 2: The ‘technology tree’ of protein design software

- Software installation can be a significant barrier for new users
- Many protein design software packages require installation of other programs and environments first
- Even if you get it installed, it may not run efficiently:
 - We saw GPU-accelerated software silently revert to using CPUs, degrading performance 10-fold



<https://www.ageofempires.com/>



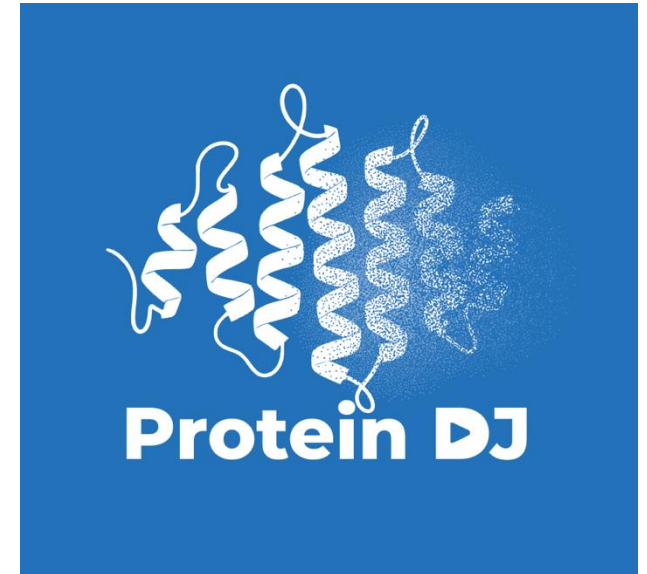
Matthew Belousoff @bluesocks81 · May 30

If one wants to de novo predict protein structures, they must first obtain master level black belt status in virtual python environments! #rfdiffusion



A vision for a new pipeline - ProteinDJ

- Easier to install and run
- Efficient in using resources
- Flexible and configurable yet consistent
- Open-source and transparent
- Informative with metadata capture and reporting
- Modular with the ability to incorporate multiple tools



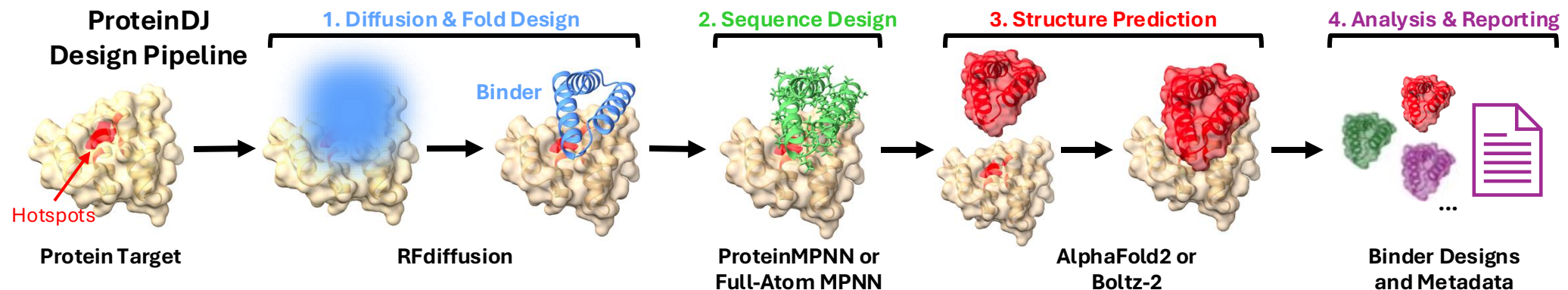
We need a tool for the present and a framework for the future

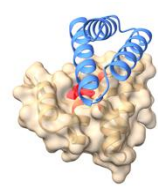
What ProteinDJ can do for you

ProteinDJ – Overview

Four main stages of the workflow:

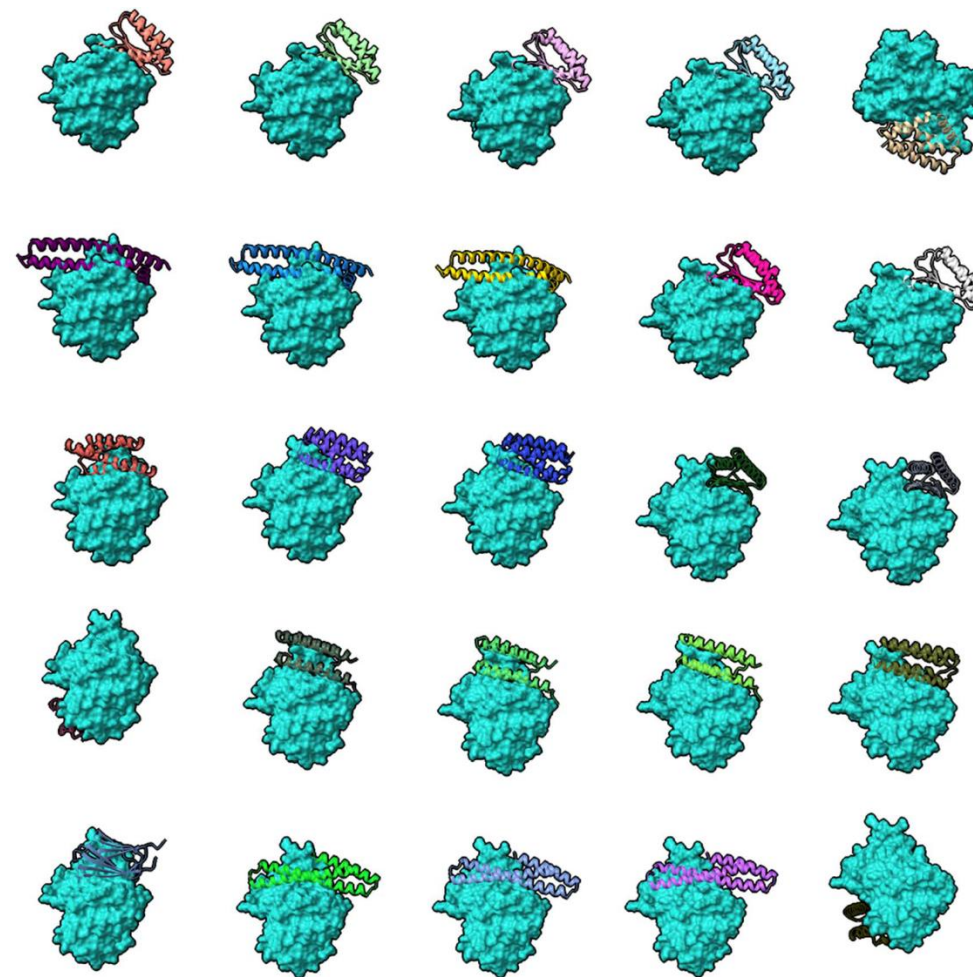
1. Fold design
2. Sequence design
3. Structure prediction
4. Analysis and Reporting

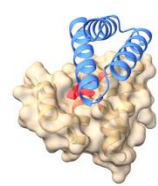




Stage 1: Fold Design

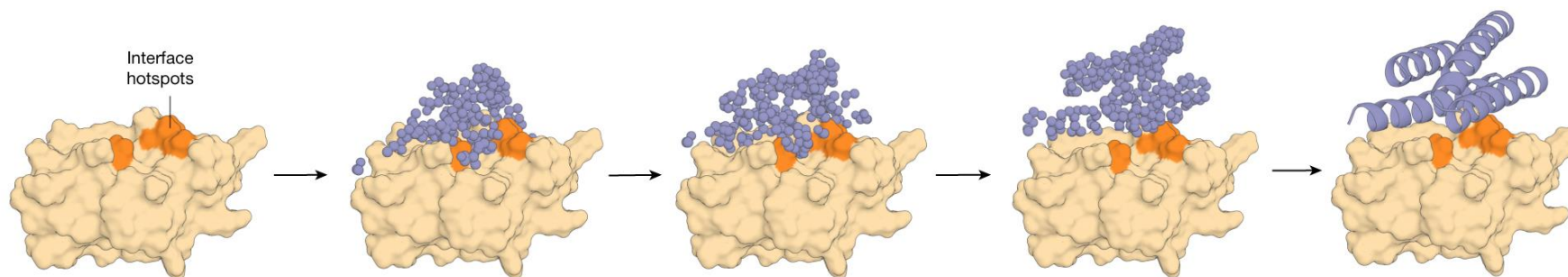
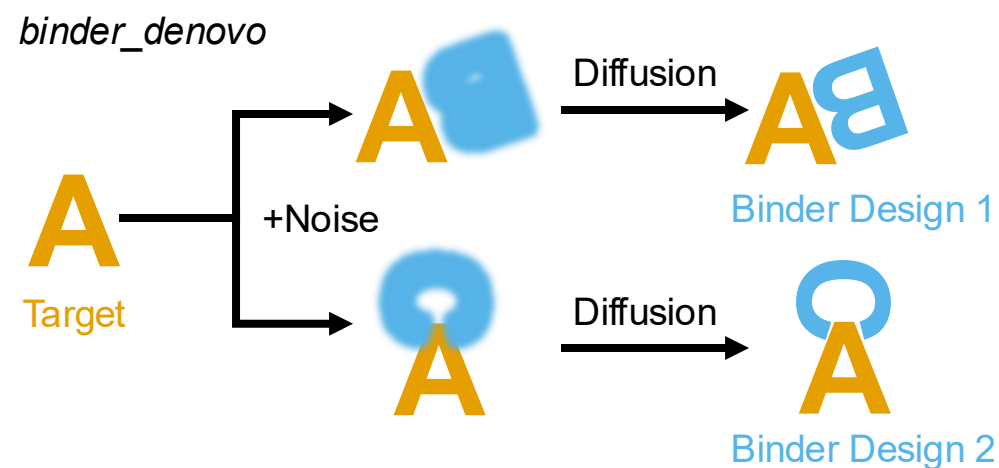
- RFdiffusion is used to generate binder folds
 - ~1-3 minutes per design
- RFdiffusion is a flexible program with many optional parameters
- To simplify it for users we created design modes for either monomer or binder design:
 - *De novo* mode
 - Fold conditioning mode
 - Partial diffusion mode
 - Motif scaffolding mode

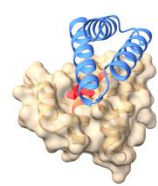




De novo mode

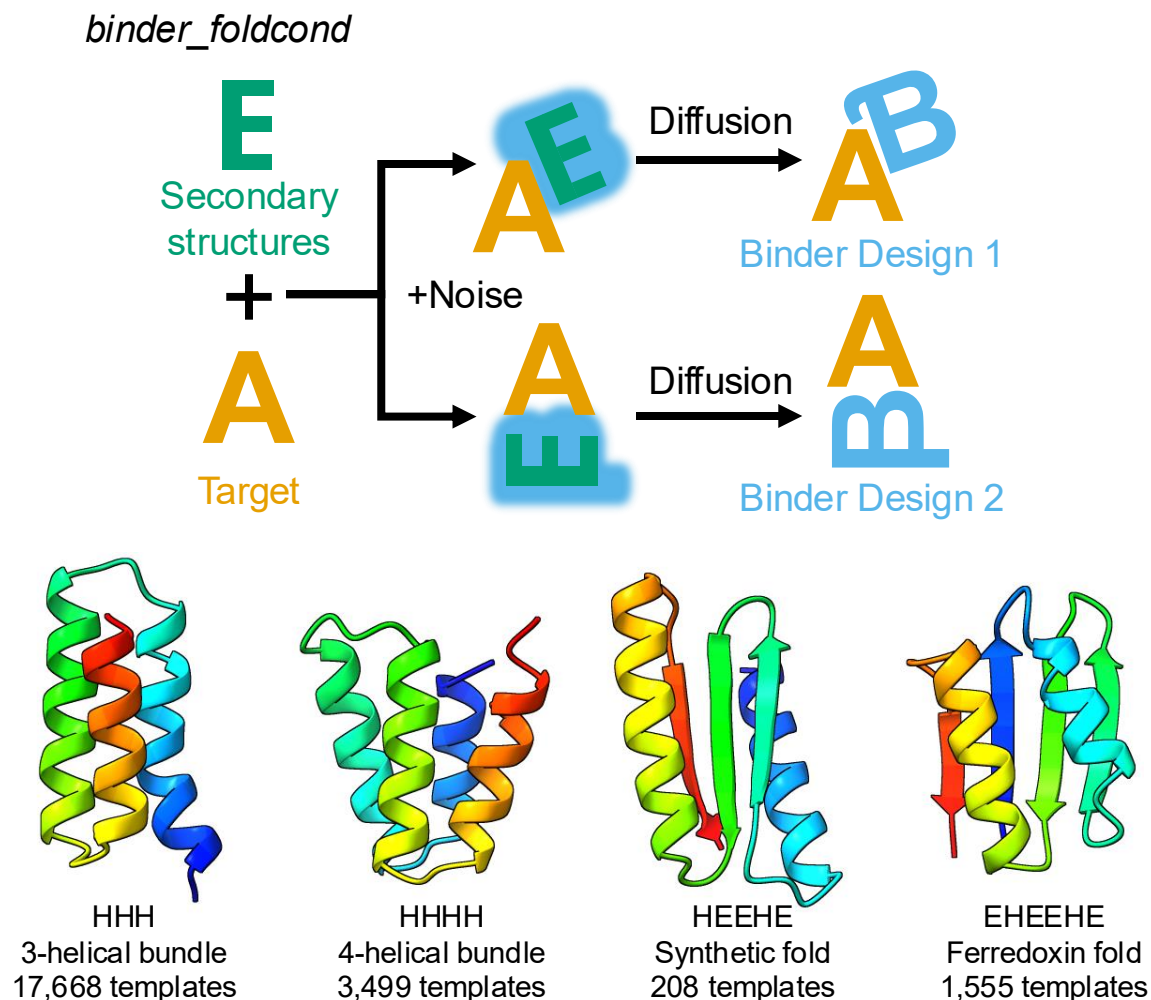
- A crowd favourite – for when you want new binders without any restrictions
- Will diffuse a binder of specified length near the target
- Can provide hotspots, we recommend trying with and without
 - It is interesting to see where RFdiffusion will prefer to diffuse a binder





Fold conditioning mode

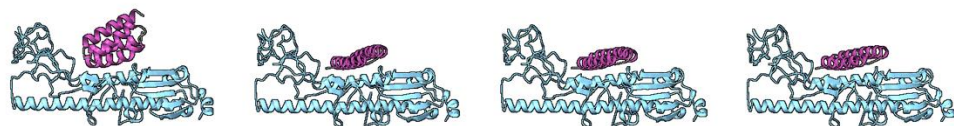
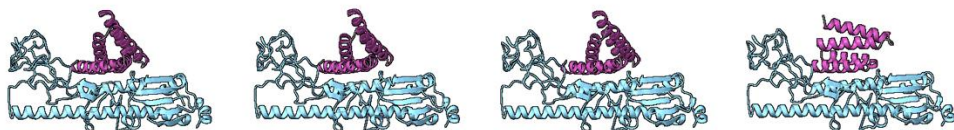
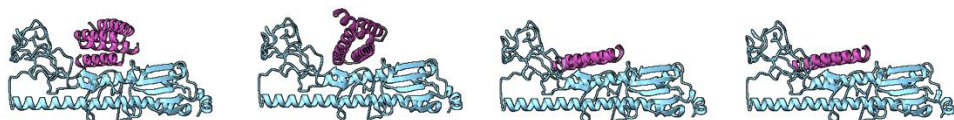
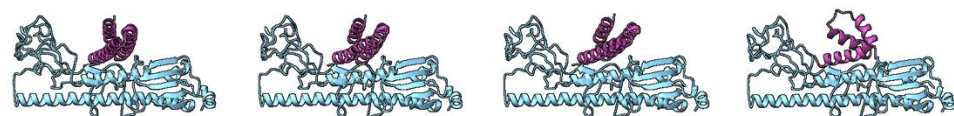
- Rather than creating ‘de novo’ binders, you can use existing folds as templates
- This was a popular approach before RFdiffusion
- Advantage in obtaining fold diversity
 - RFdiffusion has a strong bias to alpha-helical binders
- I have collated recommended fold templates published by the Baker lab on our GitHub



Fold conditioning example

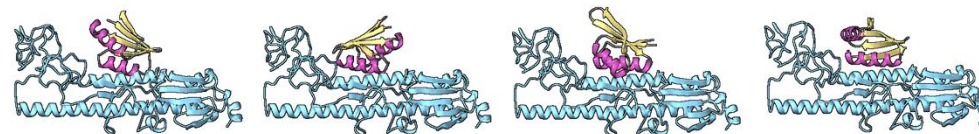
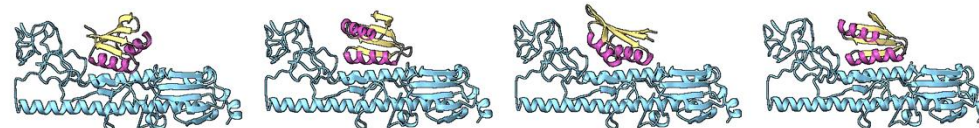
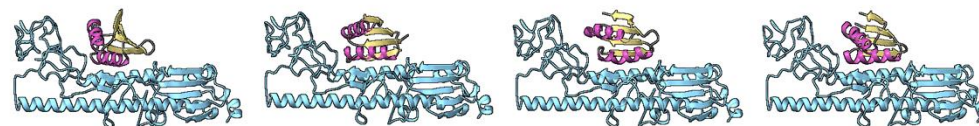
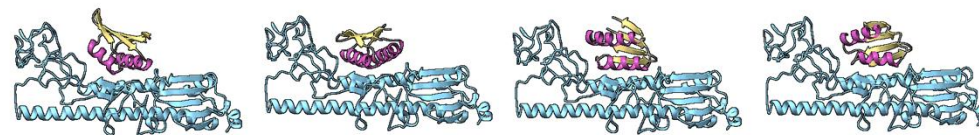
800 binders against a benchmarking target - Influenza A H1 haemagglutinin (HA)

De novo mode (only helical binders succeeded)



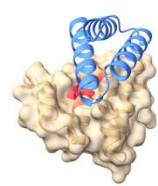
Base model success rate = 3.0%
'Beta' model success rate = 2.4%

Fold conditioning mode with Ferredoxin fold



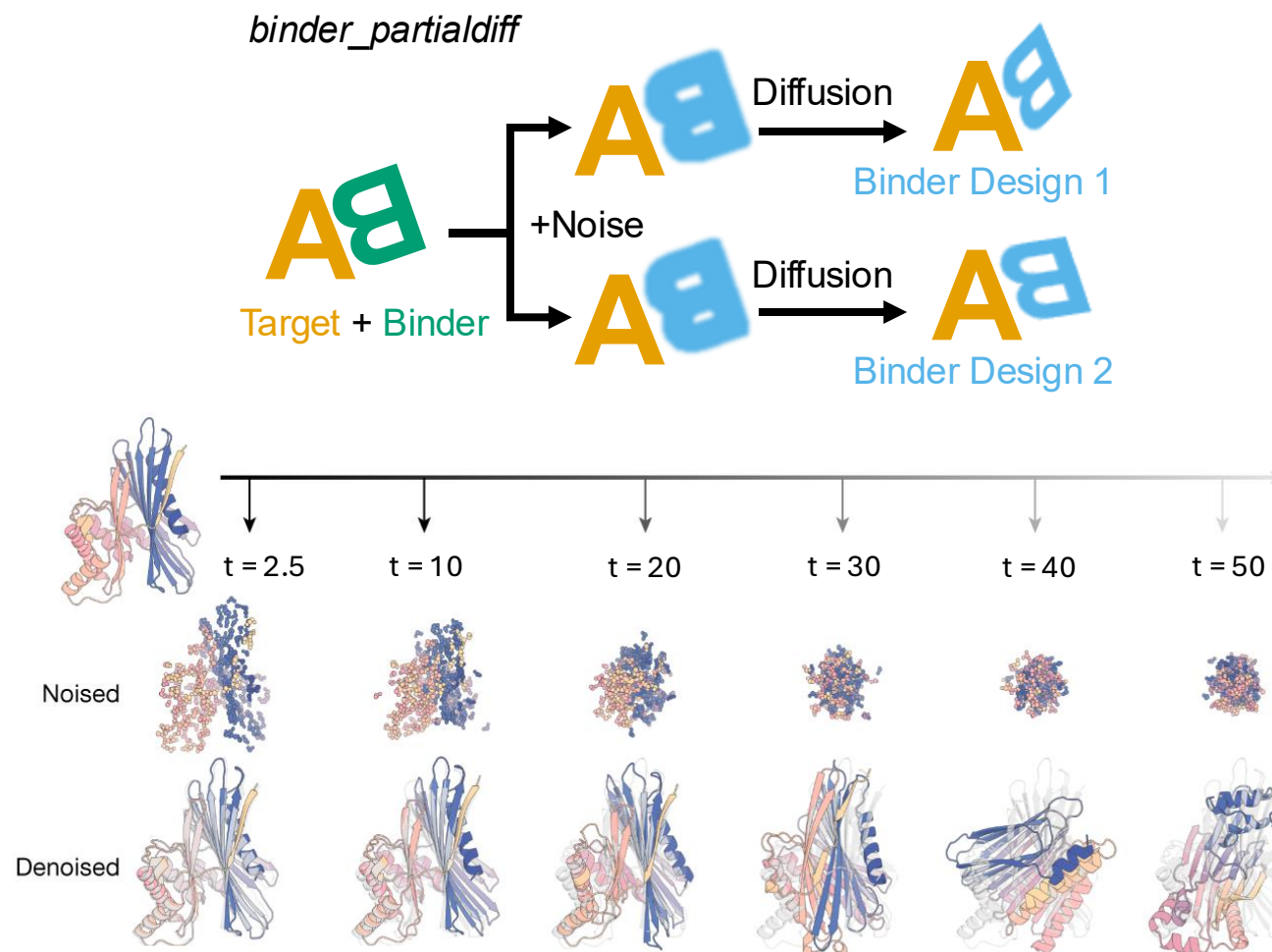
Success rate = 8.3%

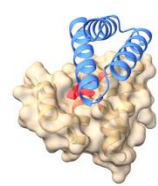




Partial diffusion mode

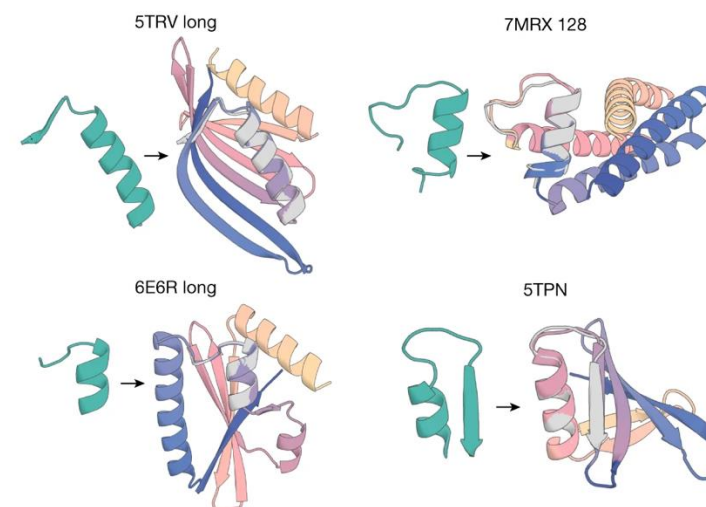
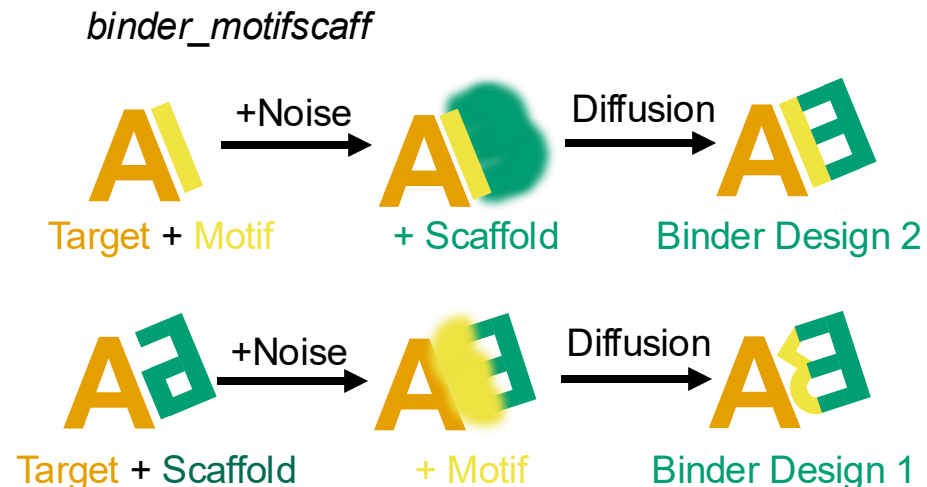
- For when you want to diversify a binder e.g. a promising de novo design that needs refinement
- Will partially noises/denoises an input fold (will design a new sequence too)
- The number of timesteps will determine how much noise will be added:
 - 10-20: Slight variations in fold
 - 30-40: Major variations in fold
 - 50: Full diffusion of fold

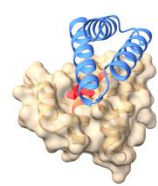




Motif-Scaffolding mode

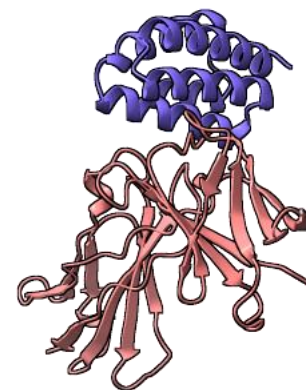
- The most complex mode
- For when you have part of a binder, either:
 - A binding 'motif' e.g. peptide
 - Or a 'scaffold' e.g. framework
- Requires careful placement of elements in input PDB file and specification of insertion residues and length





Filtering RFdiffusion outputs

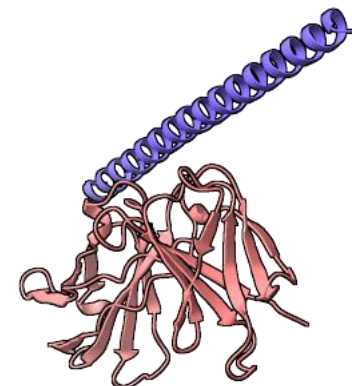
- Rfdiffusion sometimes produces low-quality or undesired binder folds
 - E.g. single alpha-helices
- We have implemented several filters to weed these out during the run:
 - Min/Max Alpha-Helices
 - Min/Max Beta-Strands
 - Min/Max Secondary Structures (Beta-Strands+Alpha-Helices)
 - Min/Max Radius of Gyration



5-helices, RoG = 11.5 Å



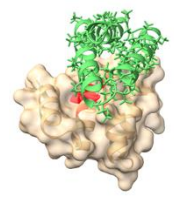
3-helices, RoG = 17.8 Å



1-helix, RoG = 25.0 Å

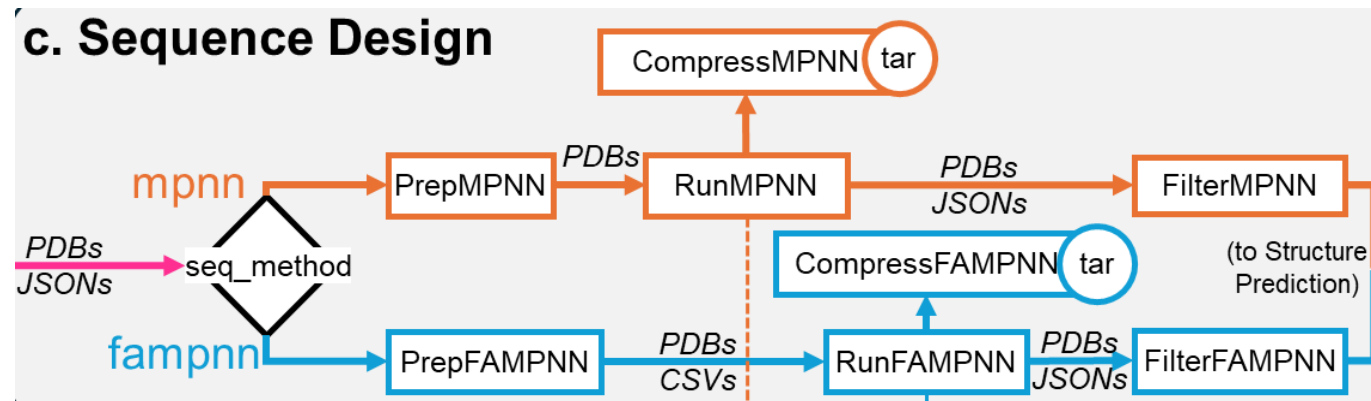


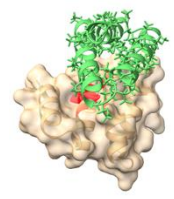
3-helices, RoG = 14.9 Å



Stage 2: Sequence Design

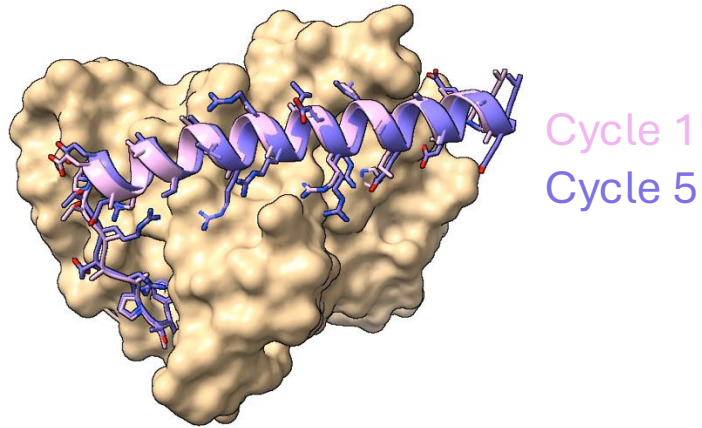
- Rfdiffusion diffuses polyglycines without sidechains or sequence information
- We need to design new sequences and side-chains, typically with 10-20 attempts per fold
- 2 options in ProteinDJ:
 - ProteinMPNN (+FastRelax)
 - Full-Atom MPNN (GPU accelerated)



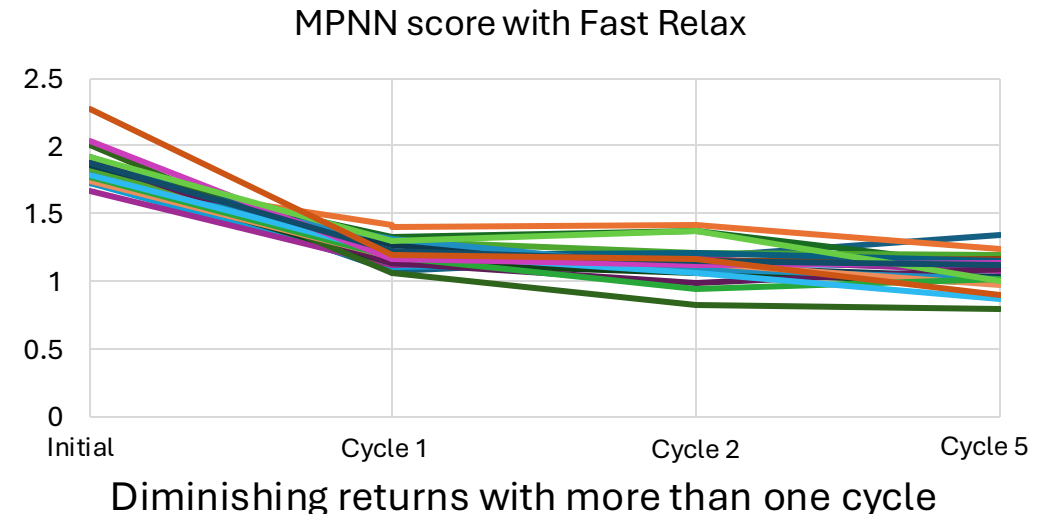


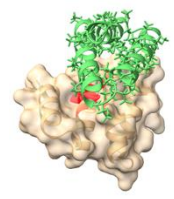
Option 1: ProteinMPNN FastRelax

- FastRelax is an optional plug-in to ProteinMPNN
- It uses Rosetta to relax the backbone and side-chains of the binder
- It is designed to be run in cycles e.g. MPNN -> FR -> MPNN -> FR -> MPNN
- It does improve success rates, but it is slow (~1-2 minutes per design) compared to a few seconds per design with ProteinMPNN only



Before: **AGYVPREATPEERAARAAASGAANAARRRARAEEAAAARA**
After: **MGFTPRERTPEERAATGAVRSRRALERRLAELAAEEEAERA**








Option 2:

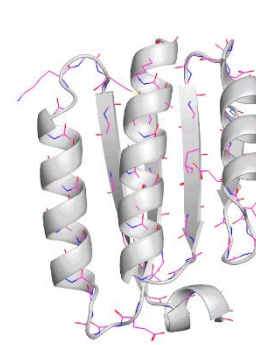
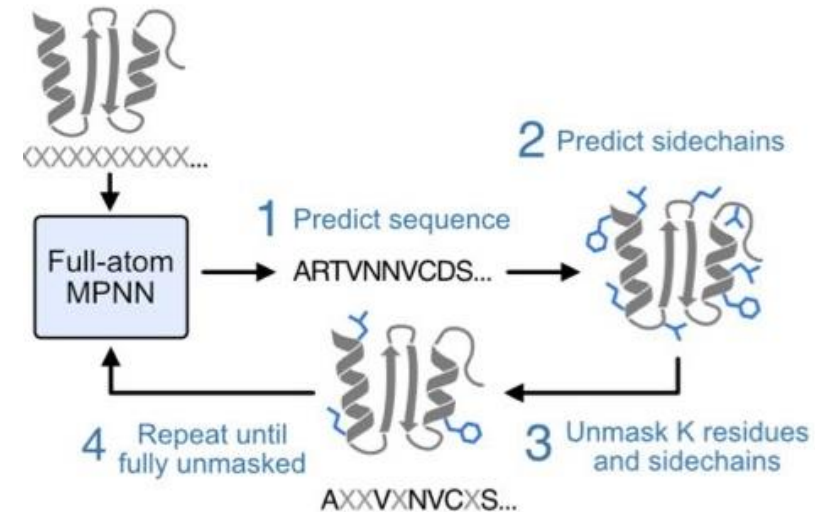
Full-Atom MPNN

- Newer algorithm that predicts sequence and side-chain conformation iteratively
- At each iteration, high-scoring side-chains are retained and it will re-attempt other side-chains
- GPU-accelerated process (~5 sec per design)
- Unlike FastRelax method, the backbone is unchanged – only side-chains atoms are refined

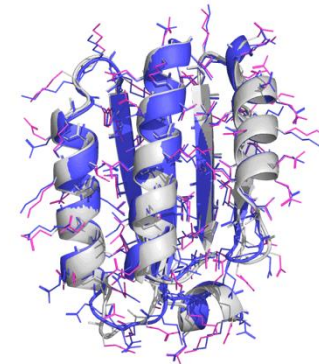
Sidechain conditioning and modeling for full-atom protein sequence design with FAMPNN

 Richard W. Shuai,  Talal Widadalla,  Po-Ssu Huang,  Brian L. Hie

doi: <https://doi.org/10.1101/2025.02.13.637498>



Sampling trajectory

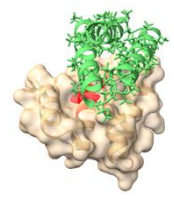


FAMPNN sample
AlphaFold2 prediction

Sequence design method comparison

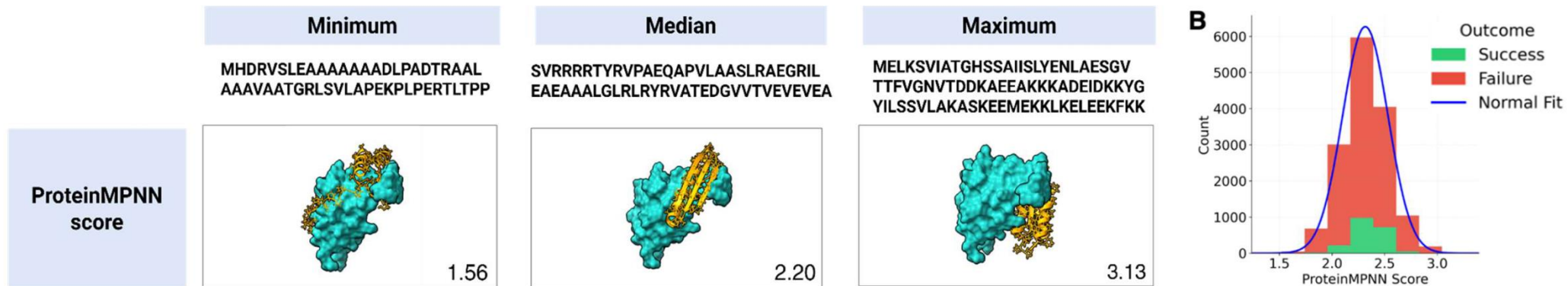
- We compared success rates for five benchmarking targets
 - Same benchmarking targets as RFdiffusion paper
- FastRelax always improved success rates
- Full-Atom MPNN was inconsistent, sometimes better than ProteinMPNN, sometimes worse

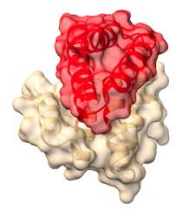
Sequence Design Method	HA	IL7Rα	IR	PD-L1	TrkA
ProteinMPNN(Vanilla)	3.9%	6.0%	21.3%	30.6%	5.4%
ProteinMPNN(Vanilla) + FastRelax	7.0%	11.1%	26.4%	47.5%	12.1%
ProteinMPNN(Soluble)	5.0%	5.8%	27.6%	33.9%	8.5%
ProteinMPNN(Soluble) + FastRelax	6.1%	9.1%	29.6%	52.3%	18.1%
FAMPNN	0.1%	10.9%	11.6%	30.9%	7.5%



Sequence design filtering

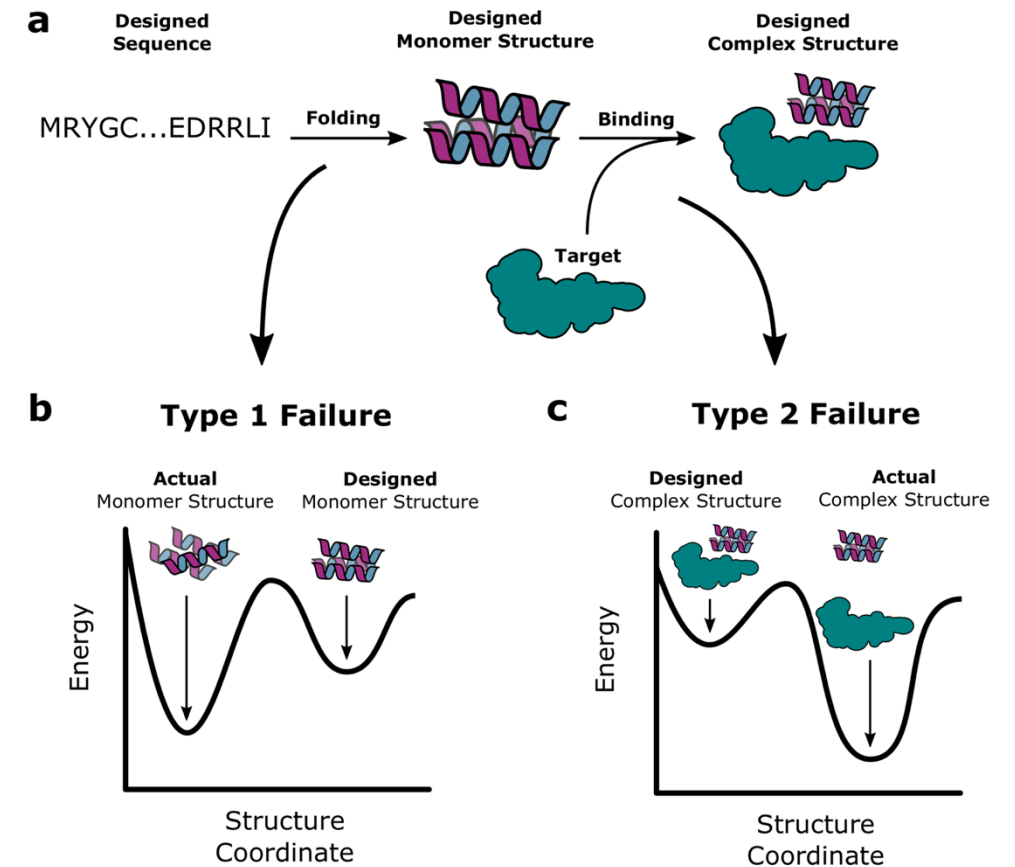
- As with the outputs of RFdiffusion, you can filter the sequence designs by score:
 - Max Negative Log-Probability (ProteinMPNN)
 - Max Predicted Side-Chain Error (FAMPNN)
- It is unclear if these values are predictive of success
 - However, it will be useful to capture and retrospectively analyse these scores and see if there are effective cut-offs

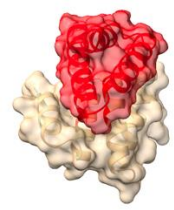




Stage 3: Structure prediction

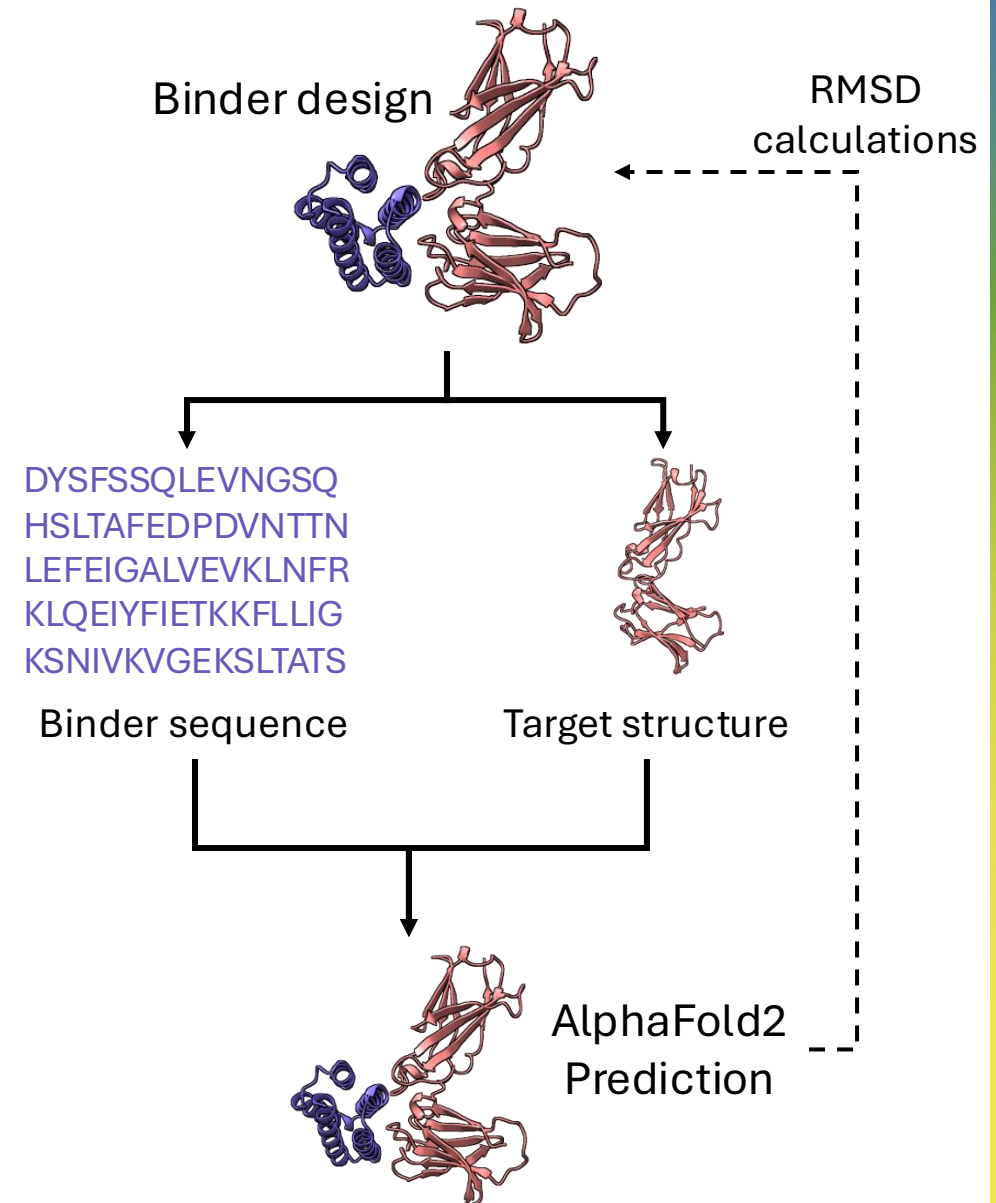
- Structure prediction of a complex between the binder and target is currently the most effective way to test designs in silico
- It is the 2nd longest computational step in the pipeline – 0.5-2 minutes per design
- 2 options in ProteinDJ
 - AlphaFold2 Initial Guess
 - Boltz2

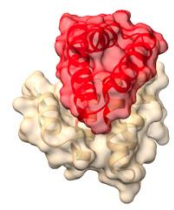




AlphaFold2 'Initial-Guess'

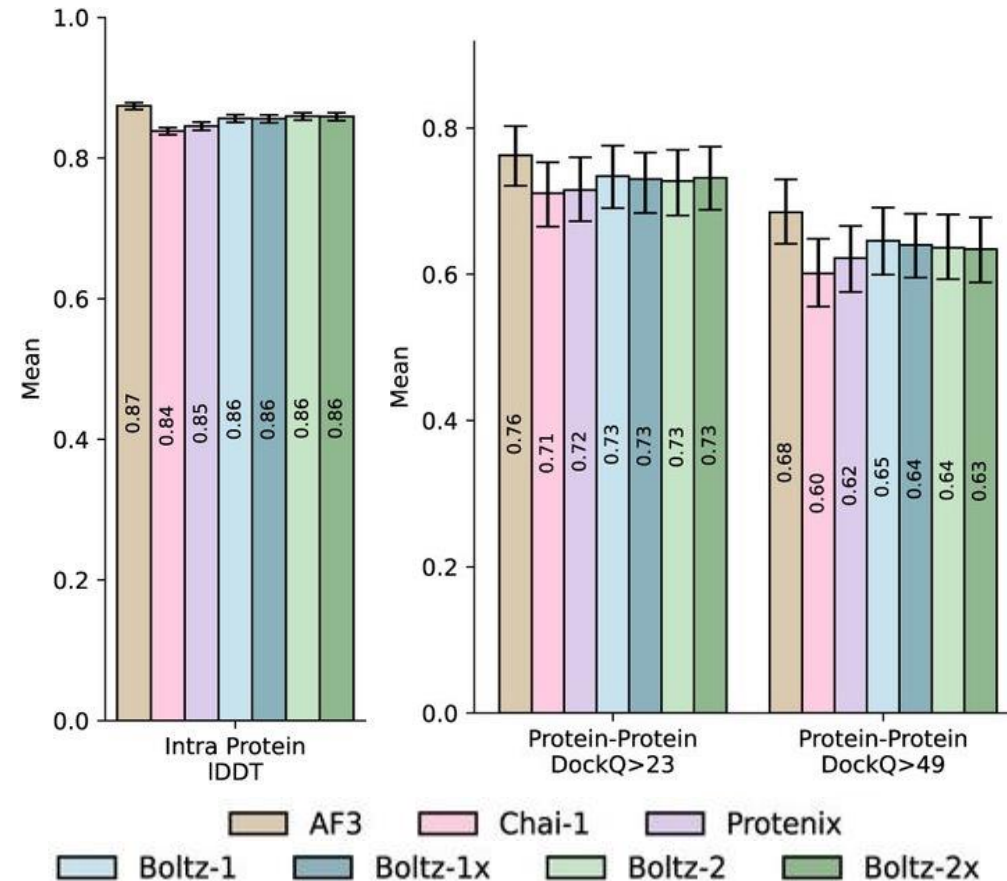
- AlphaFold2 Initial Guess is a modified version of AF2 for binder design
- AF2 is given the structure of the target (an initial guess) but only the sequence of the binder
- Skips the MSA step. Compromises accuracy for speed
- Key metrics include:
 - PAE_interaction – interface error
 - RMSD of binder between input and output













Boltz-2

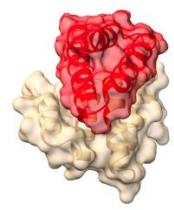
- Next generation structure prediction
 - Seeks to emulate AlphaFold3 but is open-source
- We skip the MSA step for speed - faster than AF2
- We generate additional metrics e.g. RMSD between prediction and design
- Currently has some limitations for targets with multiple chains



Boltz-2: Towards Accurate and Efficient Binding Affinity Prediction

Saro Passaro,  Gabriele Corso,  Jeremy Wohlwend,  Mateo Reveiz,  Stephan Thaler, Vignesh Ram Somnath, Noah Getz,  Tally Portnoi, Julien Roy,  Hannes Stark, David Kwabi-Addo,  Dominique Beaini,  Tommi Jaakkola, Regina Barzilay

doi: <https://doi.org/10.1101/2025.06.14.659707>



Structure Prediction Metrics and Filters

- Can filter predictions using multiple metrics including some new ones
- These filters are crucial for validation – some more than others

af2_pae_interaction	AF2 PAE for binding interface residue pairs (Å)
af2_pae_overall	AF2 PAE for all residue positions (Å)
af2_pae_binder	AF2 predicted aligned error (PAE) for binder residue positions (Å)
af2_pae_target	AF2 PAE for target protein residue positions (Å)
af2_plddt_overall	Average predicted Local Distance Difference Test (pLDDT) score for all residues
af2_plddt_binder	Average pLDDT score for binder residues
af2_plddt_target	Average pLDDT score for target residues
af2_rmsd_overall	C-alpha RMSD comparing RFdiffusion design and AF2 prediction (all chains).
af2_rmsd_binder_tgtaln	C-alpha RMSD between RFD binder and AF2 binder when target chain is aligned (Å)
af2_rmsd_binder_bndaln	C-alpha RMSD between RFD binder and AF2 binder when binder chain is aligned (Å)
af2_rmsd_target	C-alpha RMSD between RFD target and AF2 target when binder chain is aligned (Å)

boltz_overall_rmsd	C-alpha RMSD between all chains of Boltz-2 prediction and RFD design (Å)
boltz_binder_rmsd	C-alpha RMSD between binder chains of Boltz-2 prediction and RFD design (Å)
boltz_target_rmsd	C-alpha RMSD between target chains of Boltz-2 prediction and RFD design (Å)
boltz_conf_score	Confidence score for Boltz-2 prediction
boltz_ptm	The predicted template modelling (pTM) score for the Boltz-2 prediction
boltz_ptm_interface	Interface pTM-score for Boltz-2 prediction
boltz_plddt	pLDDT score for Boltz-2 prediction
boltz_plddt_interface	Interface pLDDT score for Boltz-2 prediction of the complex
boltz_pde	Predicted distance error (PDE) for Boltz-2 prediction
boltz_pde_interface	Interface PDE for Boltz-2 prediction of the complex (Å)

Recommended reading:

Predicting Experimental Success in De Novo Binder Design: A Meta-Analysis of 3,766 Experimentally Characterised Binders

Max D. Overath^{1,*,}, Andreas Rygaard^{1,2,*}, Christian P. Jacobsen¹, Valentas Brasas¹, Oliver Morell¹, Pietro Sormanni², and Timothy P. Jenkins^{1,*}

¹Department of Biotechnology and Biomedicine, Technical University of Denmark, Kongens Lyngby, Denmark

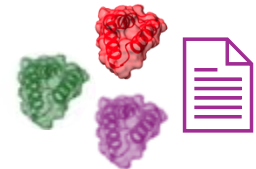
²Yusuf Hamied Department of Chemistry, University of Cambridge, Cambridge CB2 1EW, UK

*Correspondence: maxove@dtu.dk, anryg@dtu.dk, tpaje@dtu.dk

[†]Equal contribution

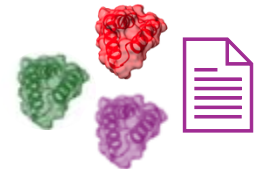
Example filters used in AlphaProteo whitepaper:

AF3		AF2 initial guess	
min_pae_interaction	< 1.5	af2_monomer_rmsd	< 1.5
ptm_binder	> 0.8	pae_interaction	< 7.0
rmsd	< 2.5	plddt	> 90



Stage 4: Analysis and Reporting

- Designs that pass structure prediction proceed to the final analysis stage
- We use PyRosetta and BioPython to generate additional metrics e.g.
 - Buried surface area
 - Delta-G of interface
 - Surface hydrophobicity
 - Isoelectric point (pI)
 - Extinction co-efficient
- These may be relevant for your use case and can provide a point of differentiation between the top designs
- Best designs are exported as PDB files and metadata as a CSV file (e.g. for use in Excel)



CSV Metadata File

- Each design has a fold ID and seq ID that matches the filename
- Easy to rank and filter in Excel

description	fold	seq	fampnn_avg_ps	af2_pae_interacti	af2_pae_over	af2_pae_binc	af2_pae_targ	af2_plddt_over	af2_plddt_binc	af2_plddt_targ	af2_rmsd_over	af2_rmsd_binder_bnd	af2_rmsd_binder_tgt	af2_rmsd_targ
fold_0_seq_0_af2pred	0	0	0.63	3.51	2.95	2.08	2.6	96.03	95.81	96.2	0.62	0.56	1.09	0.37
fold_0_seq_2_af2pred	0	2	0.73	8.02	5.43	3.15	2.77	91.42	87.65	94.37	0.81	0.66	1.61	0.35
fold_0_seq_3_af2pred	0	3	0.66	4.12	3.32	2.34	2.66	94.98	94.16	95.62	0.66	0.73	1.02	0.36
fold_0_seq_4_af2pred	0	4	0.72	3.51	2.94	2.03	2.59	96.09	96.12	96.06	0.53	0.52	0.88	0.35
fold_0_seq_6_af2pred	0	6	0.63	3.67	3.05	2.15	2.62	95.74	95.43	95.98	0.6	0.59	1	0.35
fold_0_seq_7_af2pred	0	7	0.7	3.65	3.05	2.24	2.59	95.86	95.59	96.07	0.66	0.68	1.13	0.36
fold_10_seq_0_af2pred	10	0	0.85	3.83	3.19	2.64	2.67	95.02	93.53	95.79	0.54	0.51	0.93	0.37
fold_10_seq_1_af2pred	10	1	0.82	3.71	3.12	2.58	2.64	95.38	94.19	96	0.62	0.72	1.07	0.35
fold_10_seq_4_af2pred	10	4	0.84	3.81	3.18	2.69	2.66	95.22	93.9	95.92	0.48	0.55	0.78	0.33
fold_10_seq_5_af2pred	10	5	0.82	3.98	3.3	2.84	2.72	94.85	93.36	95.63	0.46	0.49	0.73	0.35
fold_10_seq_6_af2pred	10	6	0.86	3.89	3.23	2.78	2.66	94.95	93.24	95.84	0.71	0.7	1.32	0.35
fold_11_seq_0_af2pred	11	0	0.88	4.33	3.48	2.72	2.62	93.99	91.64	95.8	1.06	0.89	2.63	0.34
fold_11_seq_3_af2pred	11	3	0.91	6.88	5.11	4.61	2.69	88.73	80.63	94.93	1.31	0.85	3.24	0.34
fold_12_seq_6_af2pred	12	6	0.73	7.38	5.01	2.74	2.71	93.37	91.81	94.57	0.92	0.39	2.24	0.34
fold_12_seq_7_af2pred	12	7	0.82	6	4.29	2.51	2.7	93.94	92.93	94.72	1.05	0.52	2.58	0.34
fold_13_seq_0_af2pred	13	0	0.76	8.83	5.74	2.71	2.76	92.98	91.07	94.46	0.79	0.49	1.95	0.36
fold_13_seq_1_af2pred	13	1	0.72	8.29	5.41	2.48	2.71	93.86	92.84	94.66	0.68	0.56	1.56	0.36
fold_13_seq_2_af2pred	13	2	0.7	8.48	5.5	2.36	2.76	93.97	93.14	94.61	0.74	0.52	1.26	0.36
fold_13_seq_3_af2pred	13	3	0.81	9.49	6.06	2.7	2.77	93.12	91.6	94.29	0.83	0.58	1.87	0.35
fold_13_seq_4_af2pred	13	4	0.83	8.6	5.57	2.45	2.74	93.47	92.34	94.35	0.56	0.45	1.15	0.35
fold_13_seq_6_af2pred	13	6	0.79	8.31	5.5	2.88	2.72	92.37	89.78	94.37	0.83	0.44	1.89	0.34
fold_15_seq_0_af2pred	15	0	0.78	4.45	3.58	3.34	2.61	94.34	91.99	95.75	0.63	0.55	1.1	0.34
fold_15_seq_1_af2pred	15	1	0.82	4.68	3.69	3.13	2.71	94.27	92.44	95.37	0.79	0.46	1.7	0.35
fold_15_seq_6_af2pred	15	6	0.82	5	3.94	3.75	2.75	93.38	90.31	95.23	0.61	0.47	1.11	0.35
fold_15_seq_7_af2pred	15	7	0.95	5.62	4.21	3.8	2.68	93.22	89.74	95.31	0.66	0.58	1.12	0.33
fold_17_seq_3_af2pred	17	3	0.8	4.84	3.8	3.03	2.77	93.61	91.04	95.3	0.85	1	1.54	0.36

Current limitations and future directions

- ProteinDJ is (currently) not suited for antibody design or design of enzymes or small molecule binders
 - These methods require specialised tools and all-atom design software e.g. RFAntibody, LigandMPNN, RFdiffusion2, ProteinDJ2?
- Bindcraft is not yet part of our pipeline and gives different binders to RFdiffusion – it's worth trying both!
- Our workflow is linear – no automatic recycling of best designs
- With new users and different HPC environments we will inevitably encounter new bugs and things we could not anticipate
 - Please log an issue on our GitHub
- With feedback and testing we hope to make the software easier to use and improve our documentation and tutorials

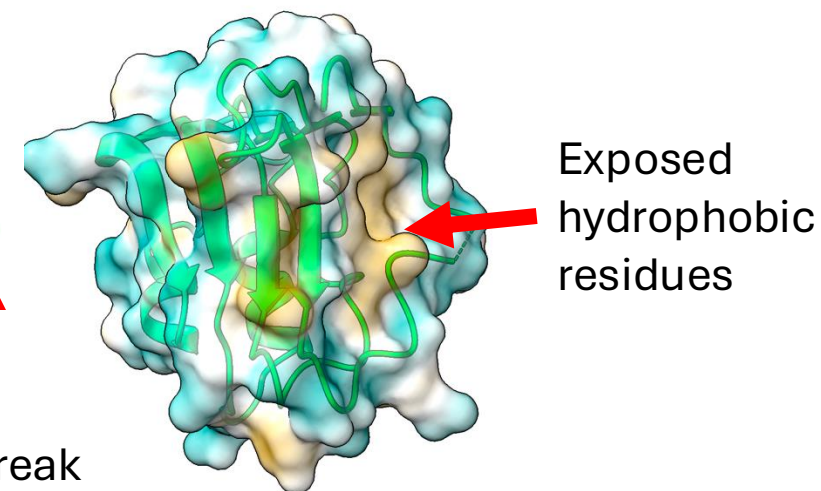
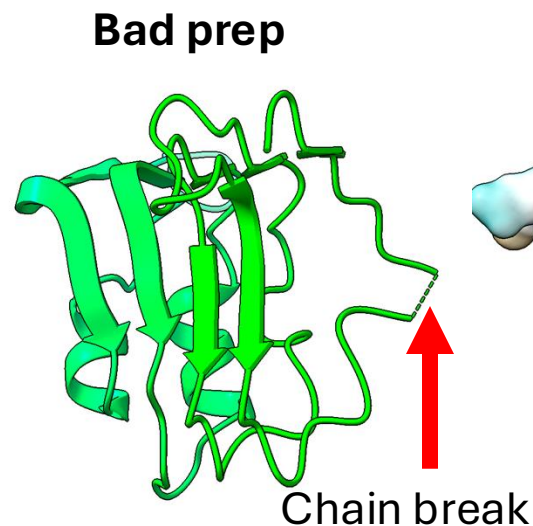
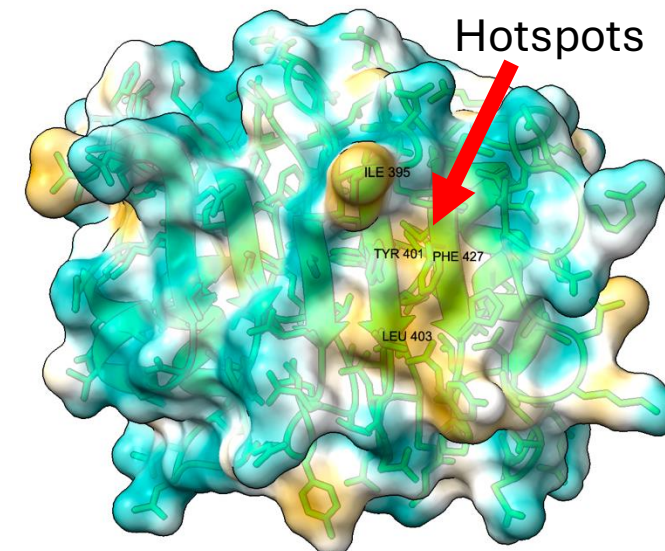
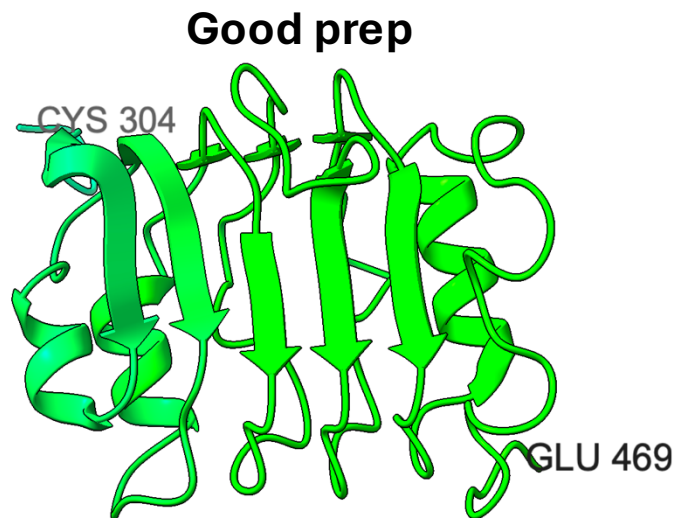
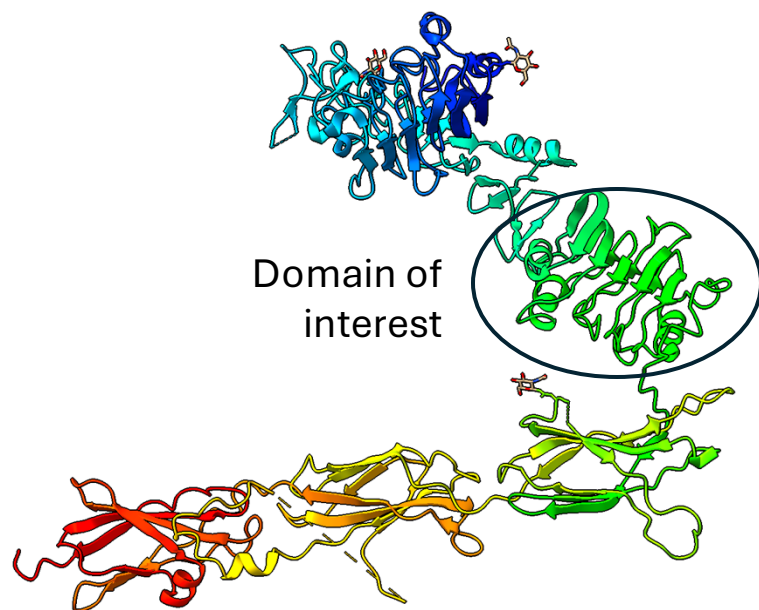
Getting started in binder design

Understanding your target protein

- Software is only part of it – understanding the structure of your target protein is essential
- Some target proteins have hydrophobic patches that RFdiffusion will easily generate a binder against – others will be more challenging
 - Try different hotspots and input models
 - Or maybe try a new homologue/target
- The process is very sensitive to the input PDB model – quality in, quality out
 - Crystal structure vs. cryo-EM structure vs. Structure prediction

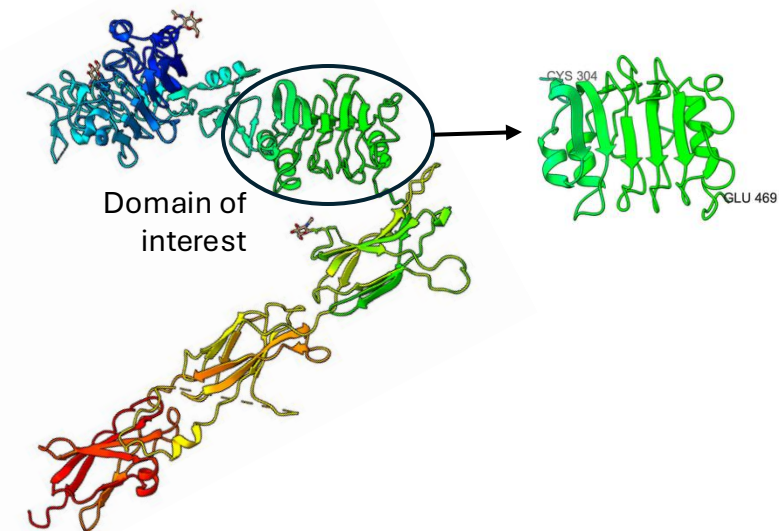
Target structure preparation example

RFdiffusion runs very slowly with large input structures so we often crop domains of interest

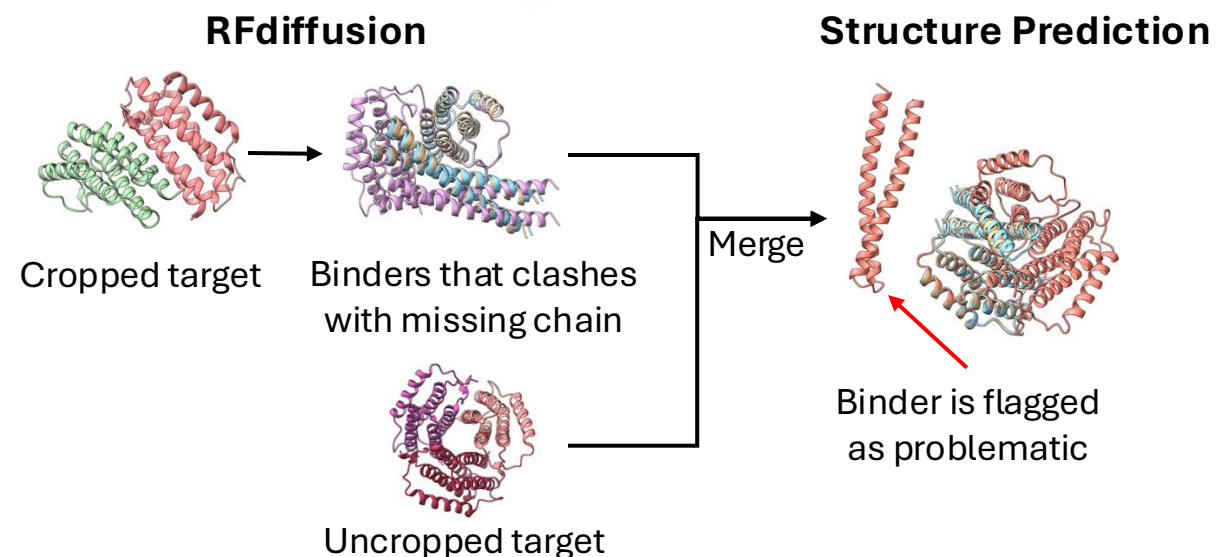


Uncropped target prediction

- Cropping target structures can expose buried interfaces and remove context
- We had added a way to provide an ‘uncropped’ target for the structure prediction step and give the prediction more context



Chains	Residues	RFD time	AF2 time
A	149	1 min	0.2 min
A + B	298	3 min	0.4 min
A + B + C	447	6 min	1 min
A + B + C + D + E + F	858	18 min	3 min



What do ‘good’ binders look like?

- Good binders come in different shapes and sizes but typically have:
 - A hydrophobic core i.e. a fold with buried side-chains
 - Shape-complementarity with the target protein
 - Sufficient buried surface area (and/or hydrogen bonds) to maintain an interaction
- Bigger is not always better – successful mini-binders can be as small as 60 residues. Try a range of sizes
- Inspect your best designs in ChimeraX and overlay with relevant structures, check for clashes with full context
- Remember it is only a prediction – your binder may not fold that way or even express at all in the lab

Identifying best parameters for binder design

- It can be difficult to guess the best parameters for binder design e.g. hotspots, length
 - And this may vary from target to target too
- This requires manual testing of parameters, which takes a lot of time and bookkeeping

Comparing MPNN checkpoints

MPNN Model	Ha	IL7Ra	IR	PD-L1	TrkA
v_48_010	3.1%	4.9%	11.6%	17.8%	2.1%
v_48_020	5.3%	5.4%	26.9%	32.3%	10.6%
v_48_030	3.9%	8.4%	26.1%	44.0%	13.5%

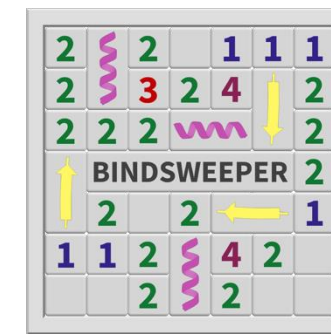
Comparing Folds for Binder Design

Program	De novo	Fold Conditioning			
		HHH	HHHH	HEEHE	EHEEHE
FAMPNN	0.5%	3.8%	3.1%	1.6%	4.1%
MPNN	3.0%	9.8%	7.8%	7.1%	8.3%
MPNNSol	3.9%	11.3%	7.1%	8.5%	7.6%

Manually generated results from 15 runs each

Bindsweeper: a parameter sweeping tool

- We developed a new tool for automated parameter sweeping
- Bindsweeper generates an N-dimensional matrix of parameter combinations and performs multiple ProteinDJ runs
- Great for testing hotspots or binder folds at small scale e.g. 100 designs, before large scale runs
- Command-line only

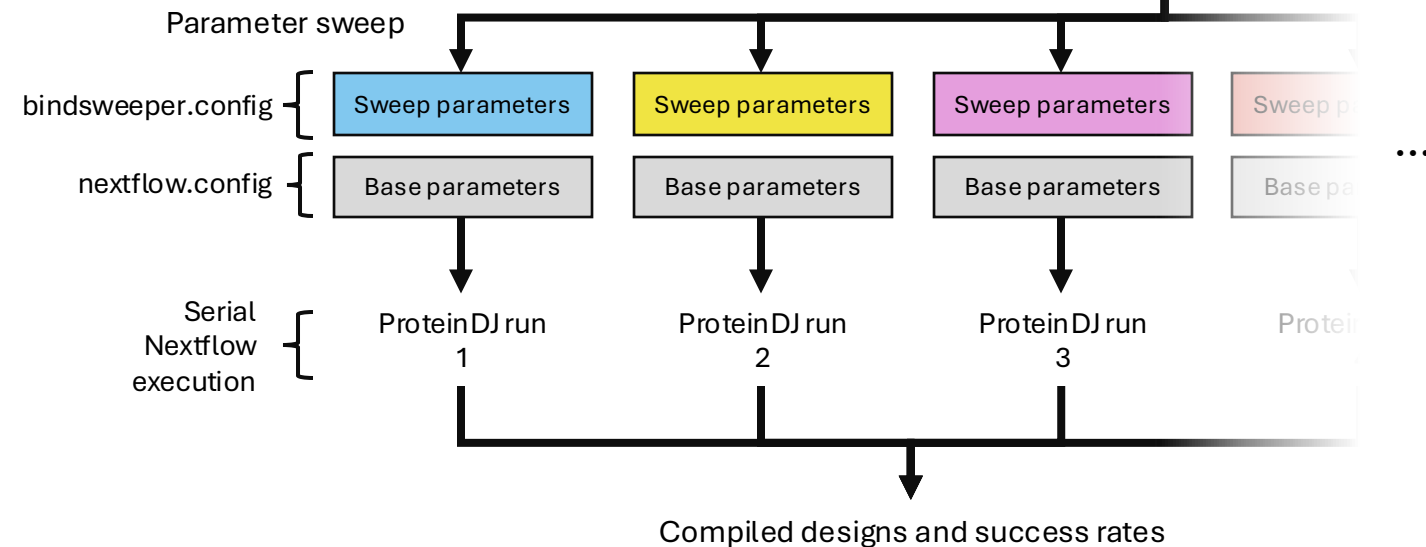


Input YAML configuration

```
fixed_params:
  rfd_ckpt_override: complex_beta
sweep_params:
  rfd_hotspots:
    values:
      - "[B13,B17,B19]"
      - "[B44, B56]"
  rfd_noise_scale:
    min: 0.0
    max: 1.0
    step: 0.5
```

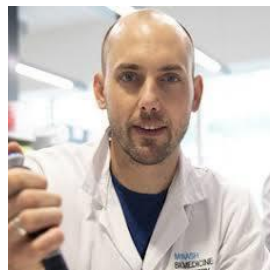
Generation of sweep matrix

Checkpoint	Hotspots	Noise scale
complex_beta	"[B13,B17,B19]"	0.0
complex_beta	"[B13,B17,B19]"	0.5
complex_beta	"[B13,B17,B19]"	1.0
complex_beta	"[B44, B56]"	0.0
complex_beta	"[B44, B56]"	0.5
complex_beta	"[B44, B56]"	1.0



Behind the Scenes: Development and Design of ProteinDJ

My first steps into the world of *de novo* protein design



Rhys Grinter



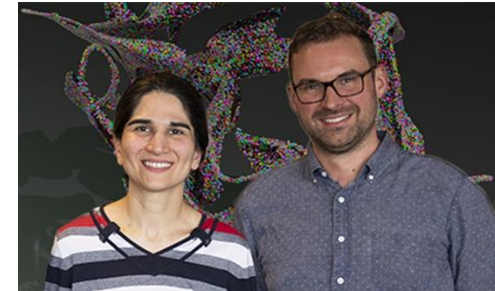
Marija Dramicanin



Gavin Knott



Cyntia Taveneau



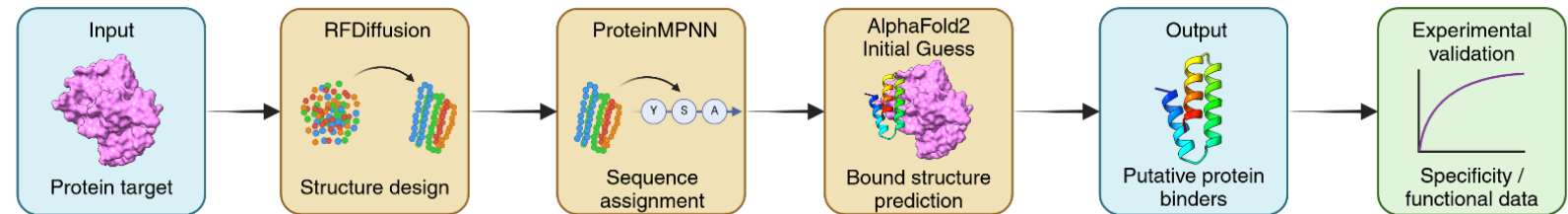
Marjan Hadian-Jazi & Dr Richard Birkinshaw

As seen previously in the BioCommons Protein Binder Seminar Series!

Inhibiting heme piracy by pathogenic *Escherichia coli* using *de novo*-designed proteins

[Daniel R. Fox](#), [Kazem Asadollahi](#), [Imogen Samuels](#), [Bradley A. Spicer](#), [Ashleigh Kropp](#), [Christopher J. Lupton](#), [Kevin Lim](#), [Chunxiao Wang](#), [Hari Venugopal](#), [Marija Dramicanin](#), [Gavin J. Knott](#) & [Rhys Grinter](#) ✉

Nature Communications **16**, Article number: 6066 (2025) | [Cite this article](#)



Hardy et al., ASBMB Newsletter 8/25

- What I found challenging getting started:
 - 3 programs/python environments to install (took me 2 weeks)
 - Mixed GPU/CPU utilisation
 - Different metadata formats
 - PDB format vs Silent format

BigScript™ – My first attempt at a workflow (June 2024)

- Workstation vs. HPC
- I wrote a BASH script that ran all three programs sequentially
 - (The only programming language I knew)
- It had some issues:
 - It only worked on WEHI HPC *and* then only in my account
 - I was requesting a GPU node, even for CPU-only steps
 - Design parameters were buried inside program commands
 - It was difficult to view intermediate results and troubleshoot
 - File organisation was a mess

Despite these limitations, it was able to generate 1000 designs in 24 hours on 4 GPUs

```
#batch RFdiffusion
export OPENBLAS_NUM_THREADS=2
export MKL_NUM_THREADS=2
for ((i=0;i<=3;i++)); do
    echo "Launching batch job on GPU ${i}"

    CUDA_VISIBLE_DEVICES=${i} ${rfdiffusionlocation}/scripts/run_inference.py \
    inference.input_pdb=${inputpdb} \
    'contigmap.contigs=[N1-121/0 400-400]' 'ppi.hotspot_res=[N9,N11,N13,N17]' \
    inference.write_trajectory=False \
    inference.model_directory_path=${rfdiffusionlocation}/models \
    inference.output_prefix=rfdresults/${outputname} \
    inference.num_designs=${numdesignsbatch} \
    inference.design_startnum=${designstartnum} \
    > logs/rfdiffusion_GPU${i}.log &

    designstartnum=$((designstartnum + numdesignsbatch)) #incrementing start number by batch size
done
wait #will not proceed until all batches are complete
echo "RFdiffusion job complete?"

#####
###Protein MPNN###
#####

micromamba activate proteinmpnn_binder_design
#convert pdbs to silent file format
silentfrompdbs rfdresults/${outputname}*.pdb > ${outputname}_rfdresults.silent

#split silent files for batching
totaldesigns=$(silentls ${outputname}_rfdresults.silent | wc -l)
batchsize=$((totaldesigns / 20))
mkdir mpnninput mpnnresults
cd mpnninput
silentsplit ../${outputname}_rfdresults.silent $batchsize
cd ..
batch=0

for inputfile in mpnninput/*; do
    mkdir mpnnresults/batch${batch}
    cd mpnnresults/batch${batch}
    ${dl_binder_designlocation}/mpnn_fr/dl_interface_design.py -silent ../../${inputfile} -relax_cycles 0 -seqs_
    batch=$((batch + 1))
    cd ../../
done
wait

#combine results into one silent file
cat mpnnresults/*.silent > ${outputname}_mpnnresults.silent
```

Bridging the gap with collaboration

- Dylan started a joint honours project with us and the Papenfuss lab, with a background in software engineering
- They proposed three software solutions:
 - Apptainer – to containerise software dependencies
 - Nextflow – to coordinate HPC resources
 - Python - to modify input/output and metadata
- A truly interdisciplinary project

Beta-
testing?

Beta-
strands?



Dylan Silke



Junqi Pan



Dr Julie Iskander



Dr Andrew Thompson



Prof Tony Papenfuss

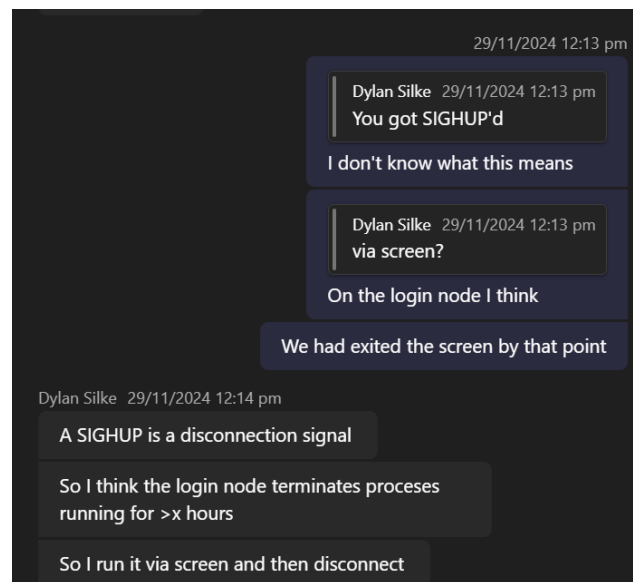


Prof Isabelle Lucet

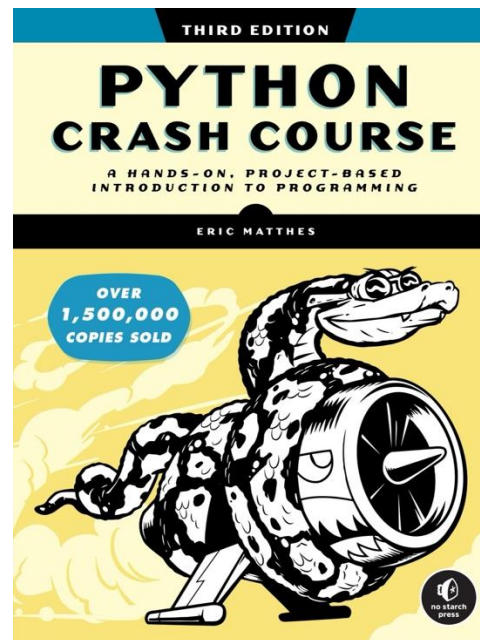
Learning how to code

- Modifying (and breaking) Dylan's scripts
- Hitting the books and documentation
- Using generative AI

Live chat with Dylan



A helpful textbook



Emotional support animal

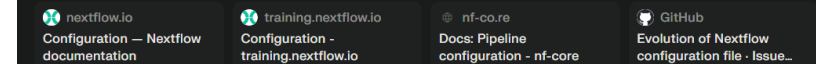


'pip install walkies'

Perplexity.AI

when using nextflow and multiple config files, how does the hierarchy of profiles work?

Answer Sources · 13 Steps



In Nextflow, when using multiple config files and profiles, the hierarchy and order determine how configuration settings and profile parameter overrides are applied. The resolution happens in two steps: config file sourcing order, and profile application order within those files.

Config File Hierarchy

- Nextflow loads configuration files in the following order, from lowest to highest priority (newest settings override older ones):
 - `$HOME/.nextflow/config` (or `$NXF_HOME/config`)
 - `nextflow.config` in the project directory
 - `nextflow.config` in the launch directory
 - Any config files specified using `-c <config-file>` on the command line
 - You can also use `-c <config-file>` to set a strict config set and ignore the defaults. `nextflow v2`

Video games



Learning to code with generative AI

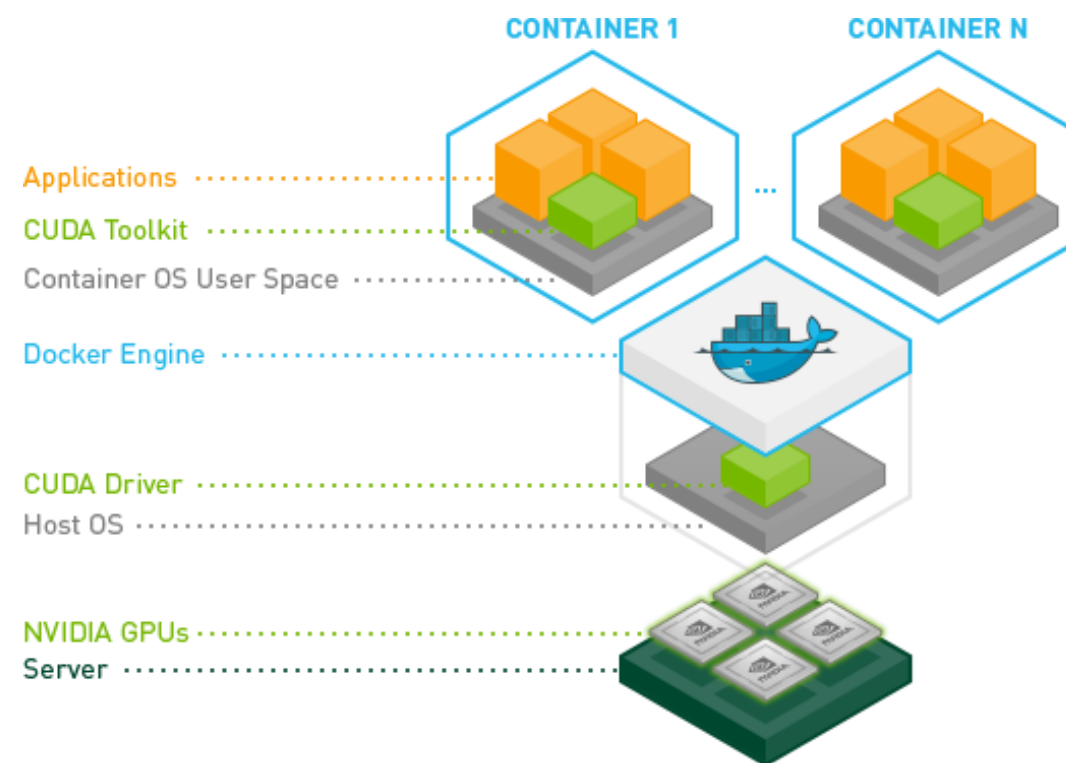
- Generative AI is really good at:
 - Understanding your intent
 - Creating clearly annotated code and functions
 - Providing suggestions for improvements and troubleshooting
- However:
 - It will assume your current code is functional and effective. It will rarely scrap what you've done and start over, even if that is more efficient!
 - You need to test everything yourself and carefully check outputs – beware well-intentioned but sometimes misleading completion messages
 - E.g. 'X completed successfully' - may not actually be true

Tips for coding with generative AI

- Not all chatbots are equal:
 - ChatGPT vs Claude
 - Reasoning/Thinking models
 - Perplexity Pro* provides access to both, easy to switch between them
- It requires a conversation, hard to succeed in a single attempt
 - Try to give as much context at the start. Better to spend 5 minutes writing a detailed prompt than an hour troubleshooting
 - Sometimes the conversation can become stale or stuck in a loop, especially when the AI becomes defensive. Try starting a new conversation and say you wrote some code and it's not very good etc.
- Learning to read the coding language helps you spot issues and provide more practical suggestions for improvements

Containers – Organ transplants for computers

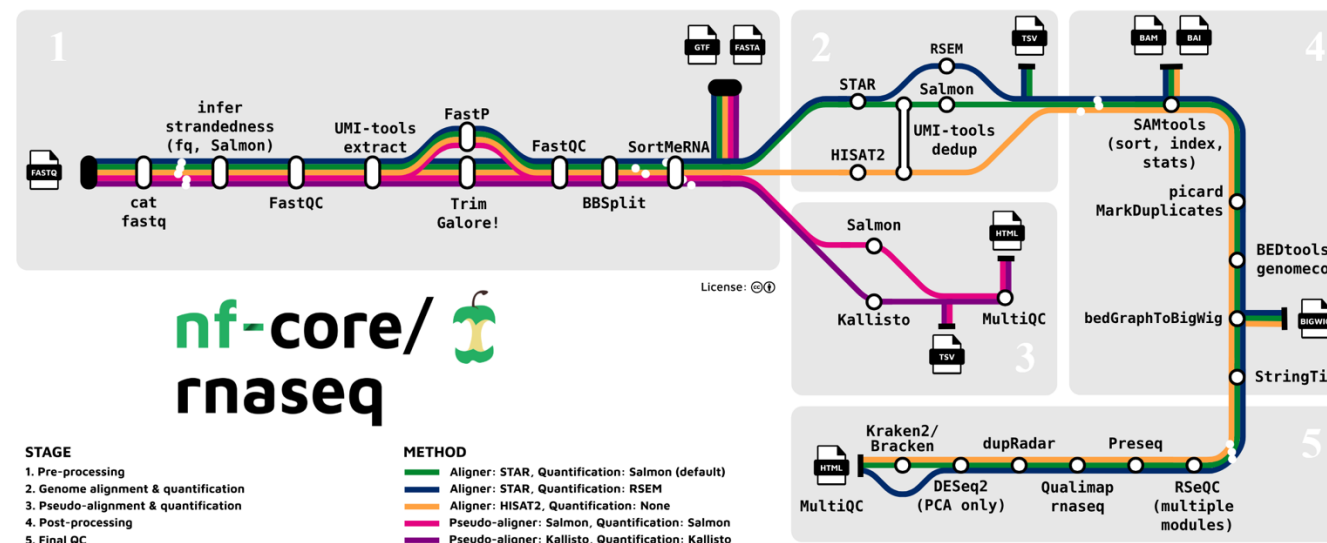
- Mini-operating systems that run within a parent operating system
- They contain the specific libraries and tools needed for a program or set of programs e.g. AlphaFold2, RFdiffusion
- Once the internal container environment is sorted (can be tricky), it will work across different computers
- They can be stored in the cloud – simply download and run



<https://docs.nvidia.com/ai-enterprise/deployment/vmware/latest/docker.html>

Nextflow – Coordinating workflows on HPC

- A Java/Groovy-based pipeline language compatible with containers and HPC (including cloud compute)
- Widely used for bioinformatics e.g. single-cell sequence data
- Data and files can be fed into ‘channels’ and tasks can be performed on each item (or batch of items)

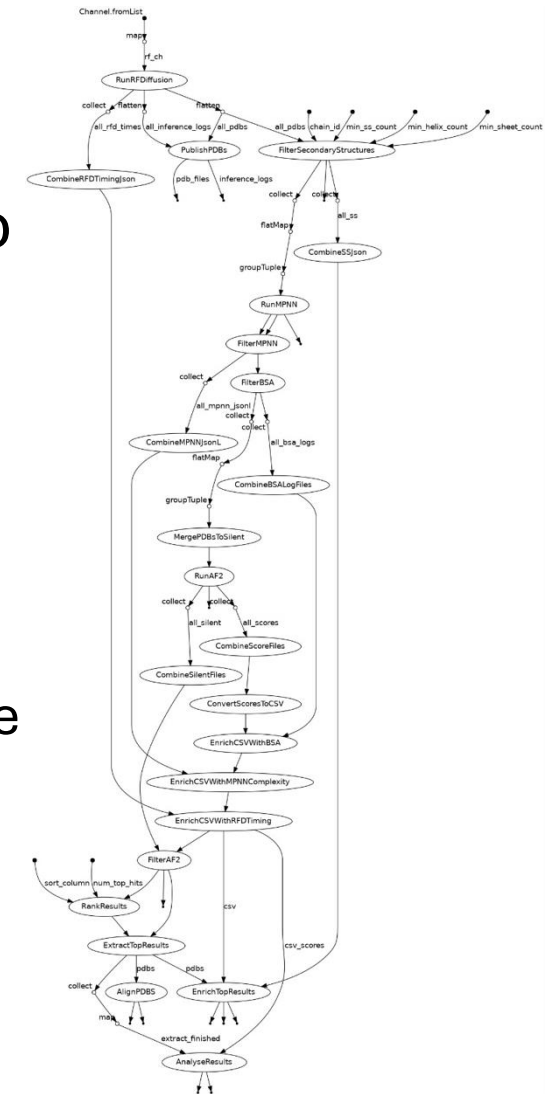


Python scripting - the glue that holds it together

- Python is a popular programming language for science and has a huge library of tools and libraries
- We used Python and the PyRosetta libraries to tweak input/output files e.g.:
 - Align two structures together
 - Extract a sequence from a PDB file
 - Calculate biophysical metrics
 - Match metadata files to PDB files
 - etc.
- Note: PyRosetta requires a license for commercial projects (but is used in Bindcraft, dl_binder_design, AlphaFold2 initial guess, etc. anyway)
 - P.S. Check out 'FreeBindCraft' for a PyRosetta workaround for BindCraft
<https://github.com/cytokineking/FreeBindCraft>

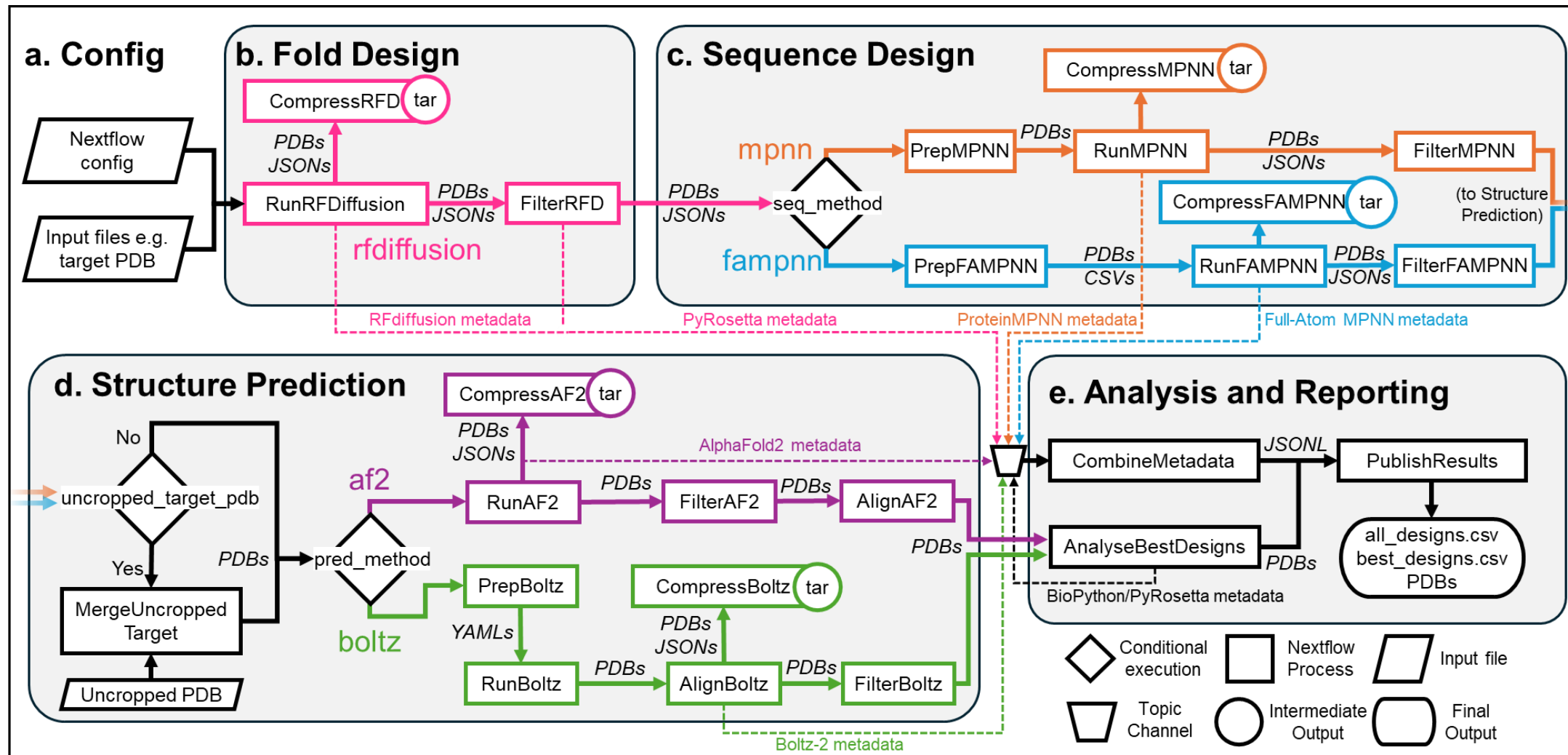
Metadata and file formats

- This was the most challenging part of development – how to get different programs to cooperate
- After several attempts we settled on the following:
 - Structure files as individual PDB files
 - Metadata files as individual JSON files
 - ID tags for folds and sequences in PDB filenames and metadata
 - Tags for metrics and parameters e.g. rfd_mode, mpnn_temperature
- Maximises interoperability and gives us a consistent start/end point for each module
 - Input: PDB files
 - Output: PDB files and JSON files



Our Modular Nextflow Pipeline

820 Commits



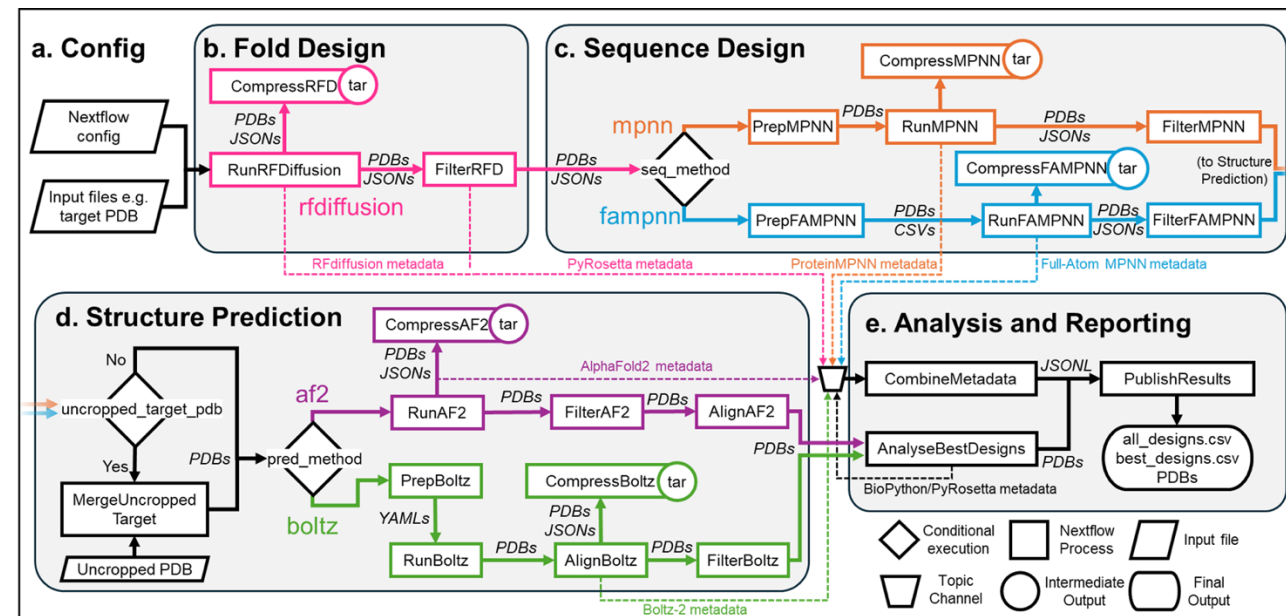
How to install and use ProteinDJ

Installing ProteinDJ

- 2 step installation process:
 1. Clone the GitHub
 - > `git clone https://github.com/PapenfussLab/proteindj`
 2. Download models/checkpoints for software (~10 mins using our script)
 - > `bash scripts/download_models.sh`
- Our containers will download automatically when you launch the software and will cache locally
 - But you can build them yourself if you prefer
- Note: If not at WEHI your HPC may have different requirements e.g. memory limits, GPU types, queue names etc. that will need to be provided to Nextflow. Ask your system admins for help.

Running ProteinDJ – Configuration

- Before running ProteinDJ you need to set the design parameters and provide any input files e.g. hotspots, target PDB files
- You can skip stages i.e. run structure prediction and analysis only
- Two ways to interact:
 - Commandline
 - Seqera (Web server)



Running ProteinDJ - Commandline

1. Edit design parameters in nextflow.config file
e.g. number of designs, input PDB
2. Run ProteinDJ
 > nextflow run main.nf
 Optionally use profiles:
 > nextflow run main.nf -profile milton,binder_denovo

The nextflow.config file

```
params {
  // ESSENTIAL PARAMETERS //
  // Pipeline mode. Choose from 'monomer_denovo', 'monomer_foldcond', 'monomer_motifscfaff'
  // 'binder_denovo', 'binder_foldcond', 'binder_motifscfaff', or 'binder_partialdiff'
  rfd_mode = binder_denovo

  // Number of designs to generate using RFdiffusion
  rfd_num_designs = 8

  // Number of sequences to generate per RFdiffusion design
  seqs_per_design = 8

  // Output directory for results. Existing results in this directory will be overwritten.
  out_dir = "${projectDir}/pdj_results"
```

Example profiles:

```
monomer_partialdiff {
  params {
    rfd_mode = 'monomer_partialdiff'
    rfd_input_pdb = './benchmarkdata/S045_pd-11.pdb'
    rfd_contigs = '[A17-111/20]'
    rfd_partial_diffusion_timesteps = 20
    mpnn_relax_max_cycles = 1
    pred_method = 'boltz'
  }
}

binder_denovo {
  params {
    rfd_mode = 'binder_denovo'
    rfd_input_pdb = './benchmarkdata/S045_pd-11.pdb'
    rfd_contigs = '[A17-131/0 60-100]'
    rfd_hotspots = '[A56,A115,A123]'
```

```
*****
PROTEINDJ
*****
ProteinDJ Protein Design Pipeline
Developers: Dylan Silke, Josh Hardy, Julie Iskander
*****
* Pipeline Mode: binder_denovo
* Number of RFdiffusion designs: 4
* Number of sequences for each design: 2
* Output Directory: test/binder_denovo
*****

The contigs for RFdiffusion [A17-131/0 60-100] include 2 chains. RFdiffusion will:
* Keep residues 17-131 of chain A
* Insert a chainbreak after chain A
* Diffuse 60-100 residues for a new chain

RFdiffusion command: /app/RFdiffusion/scripts/run_inference.py inference.write_trajectory=False inference.output_prefix=./rfd_results/fold 'contigmap.contigs=[A17-131/0 60-100]' inference.input_pdb=S045_pd-11.pdb 'ppi.hotspot_res=[A56,A115,A123]' inference.num_designs=1

[dd/f7f333] RFdiffusionWorkflow:RunRFDiffusion (B0) | 4 of 4 ✓
[fa/f8e71b] CompressRFD | 1 of 1 ✓
[ba/df0c07] FilterRFD (1) | 1 of 1 ✓
[73/48383e] PrepMPNN (1) | 1 of 1 ✓
[a5/34862a] RunMPNN (1) | 1 of 1 ✓
[e1/baa352] CompressMPNN | 1 of 1 ✓
[32/25a73c] FilterMPNN (1) | 1 of 1 ✓
[ff/95b0cc] RunAF2 (B0) | 4 of 4 ✓
[0f/df57f7] CompressAF2 | 1 of 1 ✓
[83/d000ac] FilterAF2 (1) | 1 of 1 ✓
[77/62a94e] AlignAF2 | 1 of 1 ✓
[54/e1b2b3] AnalyseBestDesigns | 1 of 1 ✓
[ae/7b8a00] CombineMetadata (1) | 1 of 1 ✓
[d2/c76d62] PublishResults (1) | 1 of 1 ✓

Pipeline results summary:
* Fold designs generated by RFdiffusion: 4
* Fold designs after RFdiffusion filtering: 4
* Sequence designs generated by MPNN (Folds * 2): 8
* Sequence designs after Sequence filtering: 8
* Predictions of designs after AF2 filtering: 8
* Final predictions for analysis: 8
```

Running ProteinDJ - Seqera Web Server

- Web interface including input form & job tracking
- Available for WEHI users through partnership with Australian BioCommons

ProteinDJ_partialdiffusion
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_motifsc scaffolding
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_denovo
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_binder_partialdiffusion
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_binder_foldconditioning
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_binder_motifsc scaffolding
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_foldconditioning
🔗 <https://github.com/PapenfussLab/proteindj>

ProteinDJ_binder_denovo
🔗 <https://github.com/PapenfussLab/proteindj>

Launch ProteinDJ_binder_denovo

1 General config — 2 **Run parameters** — 3 Advanced settings — 4 Summary

Input form view | Config view | Upload params file

Essential Parameters

These parameters are required for ProteinDJ and are used by every mode.

rfd_mode
binder_denovo

Pipeline mode.

rfd_num_designs
8

Number of designs to generate using RFdiffusion.

seqs_per_design
8

Number of sequences to generate per RFdiffusion design.

out_dir
/home/josh/pdj_binder_denovo Browse

Output directory for results.

Essential Parameters

rfd_mode *

rfd_num_designs *

seqs_per_design *

out_dir *

Mode-Specific Parameters

rfd_contigs

rfd_input_pdb

rfd_hotspots

binder_seq_method

binder_pred_method

<input type="checkbox"/>	lonely_bhaskara PapenfussLab/proteindj	PDJ_binder_denovo	✓ succeeded after 20h 16m
<input type="checkbox"/>	friendly_noyce PapenfussLab/proteindj	PDJ_binder_denovo	✓ succeeded after 17h 46m
<input type="checkbox"/>	elated_wozniak PapenfussLab/proteindj	PDJ_binder_denovo	✓ succeeded after 16h 1m
<input type="checkbox"/>	berserk_goldwasser PapenfussLab/proteindj	PDJ_binder_foldcond	✓ succeeded after 13h 19m

Processes

RFDiffusionWorkflow:RunRFDiffusion	4 of 4
CompressRFD	1 of 1
FilterRFD	1 of 1
PrepMPNN	25 of 25
RunMPNN	25 of 25
CompressMPNN	1 of 1
FilterMPNN	15 of 15
RunAF2	4 of 4
CompressAF2	1 of 1
FilterAF2	15 of 15

Aggregate stats

🕒 **13 h 19 m 6 s**
wall time

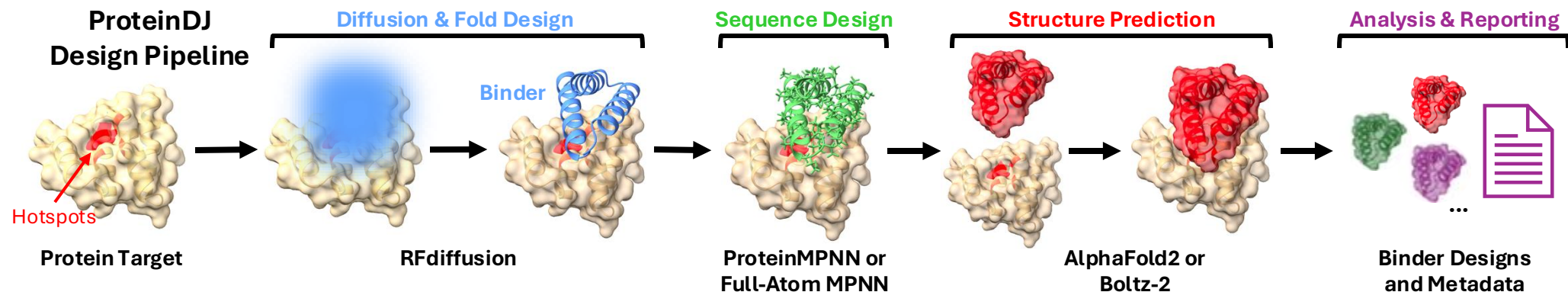
💻 **575.9 CPU hours**
CPU time

💾 **80.76 GB**
total memory

📡 **17.65 GB** **7.13 GB**
read write

Conclusion

- ProteinDJ is a tool for the present and a framework for the future
- The protein design software landscape is rapidly evolving and there is a need for conventions and interoperability
- We have made our code open-source to enable collaborative efforts to build similar workflows and incorporate future software



Acknowledgements

Dylan Silke

Isabelle Lucet

Andrew Thompson

Junqi Pan

Emily Park

Lyn Deng

Lurlene Trepout

Julie Iskander

Tony Papenfuss

***De novo* Protein
Design Community**

Richard Birkinshaw

Marjan Hadian-Jazi

Daniel Brown

Marija Dramicanin

David Zhu

Rhys Grinter

Cyntia Taveneau

Gavin Knott








Kate Michie

Johan Gustafsson



bioRxiv Preprint:

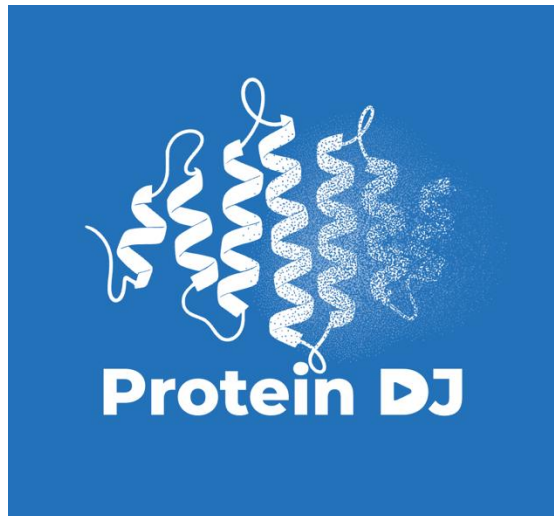
ProteinDJ: a high-performance and modular protein design pipeline

 Dylan Silke,  Julie Iskander,  Junqi Pan,  Andrew P Thompson,  Anthony T Papenfuss,
 Isabelle S Lucet,  Joshua M Hardy

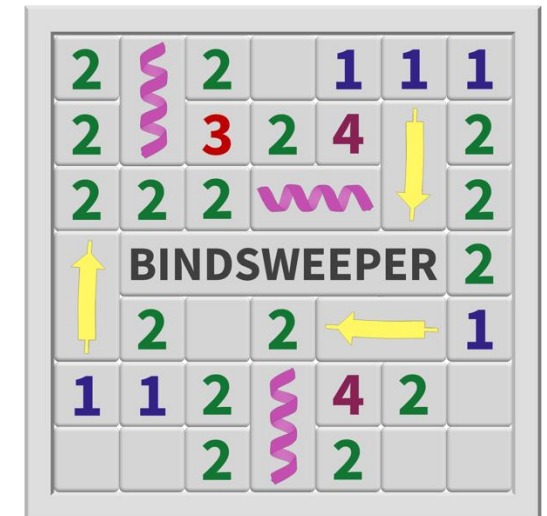
doi: <https://doi.org/10.1101/2025.09.24.678028>

GitHub: <https://github.com/PapenfussLab/proteindj>

WEHI Research Computing



Any questions?



Socials/Links