Dry Lab Notebook

Submodel Point Mutations

iGEM Lund Trashformers

Table of Contents

[Template] 2025-XX-XX Title

Contributors:

Objective

What is the purpose of this session?

Background

Is it based on any articles, research etc. Cite!
Or is it based on previous work we have done? Refer clearly to at least which entry nbr and the title of that entry!

Methods and parameters

Component	*Description*
Software Used	
Input Structure	
Key Parameters	
Script Location	

Code

To insert Code Block: Go to Add-ons → Code Blocks
This is a guide: https://www.geeksforgeeks.org/how-to-add-code-block-in-google-docs/

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

Interpretation

Next Step

2025-06-07 Notes: On how to perform ligand dockings

Contributors: Sixten

Objective

Starting research on how to do docking. Two main strategies: Look for papers and investigate workflow/programs used; Use AI to get an overview of the subject. Trying a machine-learning based method to dock substrate to acyl-activating enzymes.

Overview/Background

Of the numerous molecular docking tools available, several are widely recognized for their application in structure-based drug design. Prominent software packages include AutoDock and AutoDock Vina, Schrödinger's Glide, Molecular Operating Environment (MOE), GOLD, and RosettaDock. The successful application of these tools is contingent upon the careful preparation of several key inputs.

The primary input is the three-dimensional structure of the target enzyme, typically derived from a Protein Data Bank (PDB) file. This structure requires significant preprocessing to ensure accuracy. This preparation involves the removal of extraneous water molecules, unless they are mechanistically critical for ligand binding. Subsequently, hydrogen atoms must be added, with particular attention paid to polar hydrogens. It is essential to assign correct protonation states to amino acid residues, a pH-dependent process that can be accomplished using tools such as PDB2PQR, H++, or PropKa. Finally, any non-essential ligands or ions should be removed, while strategically important cofactors or metal ions are retained.

The small molecule ligand also requires meticulous preparation. The ligand structure, often obtained from databases like PubChem or ZINC or drawn in applications like ChemDraw, must be converted into an appropriate 3D format such as .mol2, .pdbqt, or .sdf. Its preparation includes generating a 3D conformation, assigning the correct protonation state and tautomeric form, and performing a geometry optimization using energy minimization methods (e.g., MMFF94). Tools like Open Babel or RDKit are frequently used for these tasks and for final conversion into the specific file format required by the chosen docking software.

A critical step is defining the precise location on the protein where docking will occur. The binding site can be identified using the coordinates of a known ligand from a co-crystal structure, by specifying key active site residues, or by employing predictive algorithms like CASTp, FPocket, or Schrödinger's SiteMap. For grid-based docking programs, this site is defined by setting the coordinates and dimensions of a grid box that encompasses the binding pocket.

Finally, all prepared structures must be converted into the precise file formats mandated by the docking software. For instance, AutoDock Vina requires inputs in the PDBQT format, which can be generated using AutoDockTools (ADT), while Glide utilizes the Maestro (.mae) format. Furthermore, if the enzymatic mechanism relies on metal ions (e.g., Zn^{2+}) or organic cofactors (e.g., NADH), these must be retained within the protein structure and their parameters correctly defined to ensure accurate simulation of their interactions with the docked ligand.

[Text generated by ChatGPT and DeepSeek]

Results/Discussion

- Google scholar was used to find research papers with the input: “molecular docking enzyme”. The following articles were found and evaluated:
    - Cellulase enzyme: Homology modeling, binding site identification and molecular docking
        - They used FlexX for docking.
- Downloaded AutoDock vina
- Downloaded the pdb file for 1T5H on UniProt: https://www.uniprot.org/uniprotkb/Q8GN86/entry#structure
- Downloaded the sdf file for the ligand 4-chlorobenzoate from Pubchem: https://pubchem.ncbi.nlm.nih.gov/compound/4359436#section=3D-Conformer

Next Steps

- Download MLGtools to edit pdb files, but have to wait until wifi is accessible. Then proceed to perform the docking.

2025-06-08 Notes: On how to perform ligand dockings 2

Contributors: Sixten

Objective

Trying methods (PyRx) to dock substrate to acyl-activating enzymes.

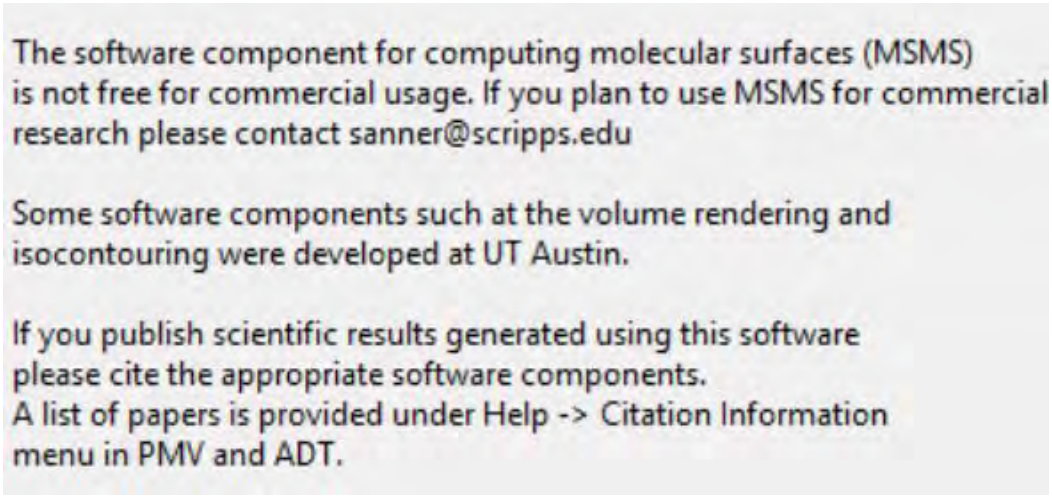
Overview/Background

The following are the general steps involved in molecular docking:

1. Preparation of the ligand: The ligand molecule needs to be prepared before docking. This involves obtaining the 3D structure of the ligand, optimizing its geometry, adding hydrogen atoms, and assigning partial charges.
2. Preparation of the protein: The target protein structure needs to be prepared for docking. This involves removing any existing ligands or water molecules, optimizing the protein structure, adding hydrogen atoms, and assigning partial charges.
3. Grid generation: A grid of points is generated around the protein's active site. The purpose of this grid is to define the sampling area for the ligand during docking.
4. Ligand placement: The ligand is placed within the defined grid space near the protein's active site. Various algorithms are used to explore different conformations and orientations of the ligand.
5. Scoring: A scoring function is used to evaluate the fitness of each ligand conformation within the binding site. This function calculates the interaction energy between the ligand and protein, considering factors such as hydrogen bonding, electrostatic interactions, and van der Waals forces.
6. Refinement and optimization: The top-ranked ligand conformations are subjected to refinement techniques like molecular dynamics simulations or energy minimization to improve their accuracy and stabilize the interactions.
7. Analysis and visualization: The docking results are analyzed and visualized to understand the binding mode and the most promising interactions between the ligand and protein. This may include examining hydrogen bonds, hydrophobic contacts, and other important features.
8. Validation: The docking results are validated by comparing them to experimental data, if available. This can involve studying the binding affinity, binding modes, and predicted binding energies.
9. Iterative optimization: If the docking results are not satisfactory, the process can be iterated by modifying or optimizing different parameters such as the grid size, scoring functions, or ligand/protein preparation protocols.

Results/Discussion

- Downloaded MLGtools. Note:



    - Do not now if this applies to us but we will publish results at least
- After failure to launch due to problems with graphics AMD card, PyRx was downloaded instead. https://www.researchgate.net/post/Cant_run_AutoDock_tools_in_Windows_11
- Realized that PyRx uses Autodock vina automatically so can do everything from there.
- Read the introduction to molecular docking on https://www.biosolveit.de/application-academy/introduction-to-docking/
    - Explained a bit of the scoring system and why docking is used but no in depth explanations
- Found a pretty good step-step tutorial (https://biotechworldindia.in/molecular-docking-a-short-overview-and-steps-involved/#google_vignette):
- Try to prepare the ligand and protein in PyRx according to the steps above.
- Prepared macromolecule of 1t5h using PyRx macromolecule function
- Prepared ligand of 4-chlorobenzoate using PyRx ligand molecule function
- Ran vina in PyRx and got results.
- Had problems with downloading results and is a bit confused on how to do this.

Next Steps

- Troubleshoot the problem with PyRx or find another method for docking.

1T5H MPNN1 constructs:2025-06-09 Docking of molecules with Rowan

Contributors: Hilya

Objective

Trying a machine-learning based method to dock substrate to acyl-activating enzymes.

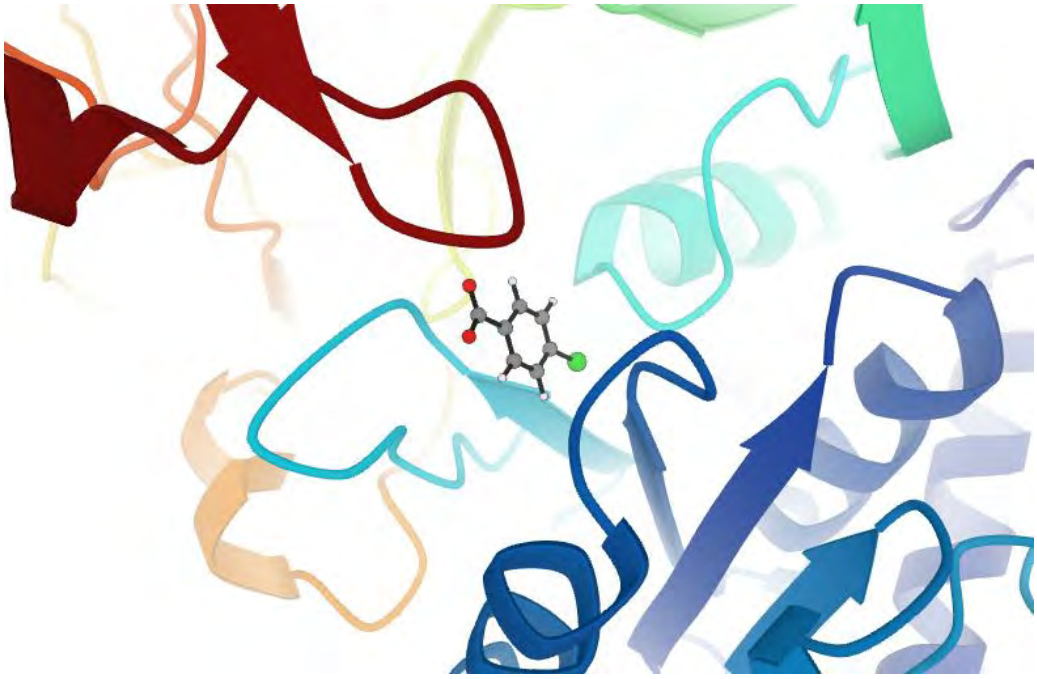
Background

We have chosen some acyl-activating enzymes as candidates to work with in our dry lab pipeline. One of them is 1T5H, 4-Chlorobenzoyl-CoA Ligase/Synthetase. Rowan is an online computational tool based on machine learning models that can be used for predicting protein-ligand interactions. Rowan's docking workflow is a combination of AutoDock Vina to generate poses and then performing a fast ligand-strain energy calculation.

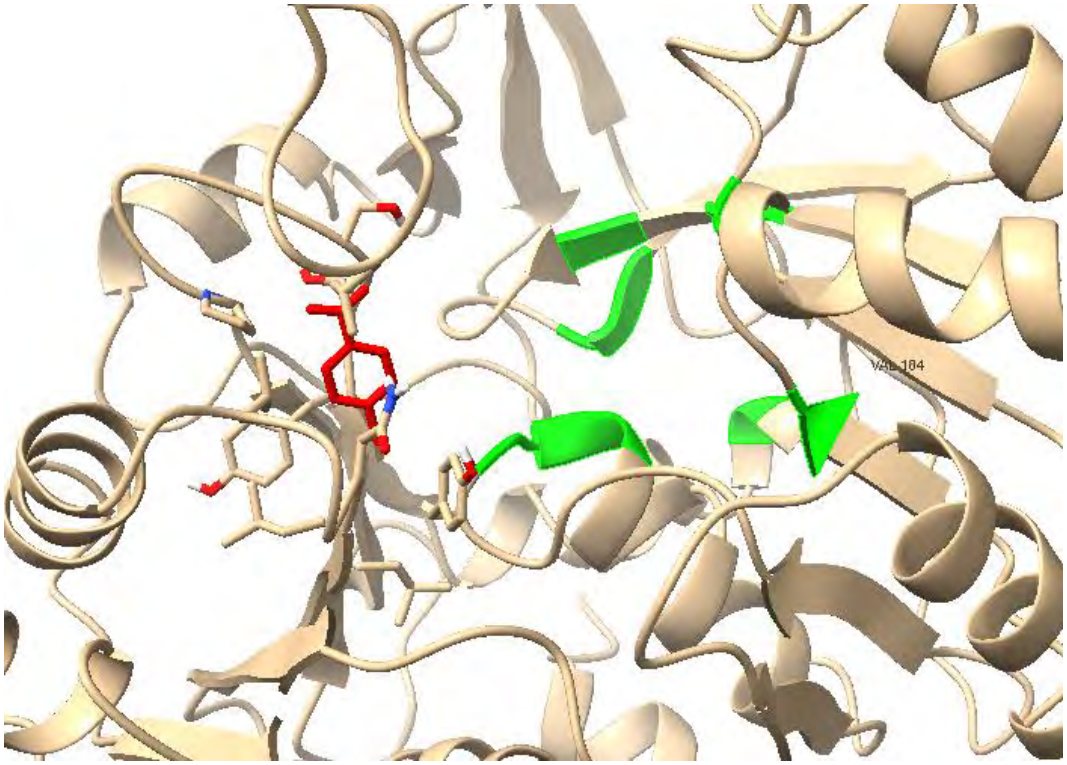
Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), ChimeraX
Input Structure	PDB file for 1T5H: RCSB PDB - 1T5H: 4-Chlorobenzoyl-CoA Ligase/Synthetase unliganded, selenomethionine 4-Chlorobenzoate 3D file: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used

- Made an account and chose drug discovery > docking on the main menu.
- Files: PDB for 1T5H from RCSB PDB - 1T5H: 4-Chlorobenzoyl-CoA Ligase/Synthetase unliganded, selenomethionine
- 4-chlorobenzoate form 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
- Sanitised the 1T5H using the built-in option
- Performed blind docking



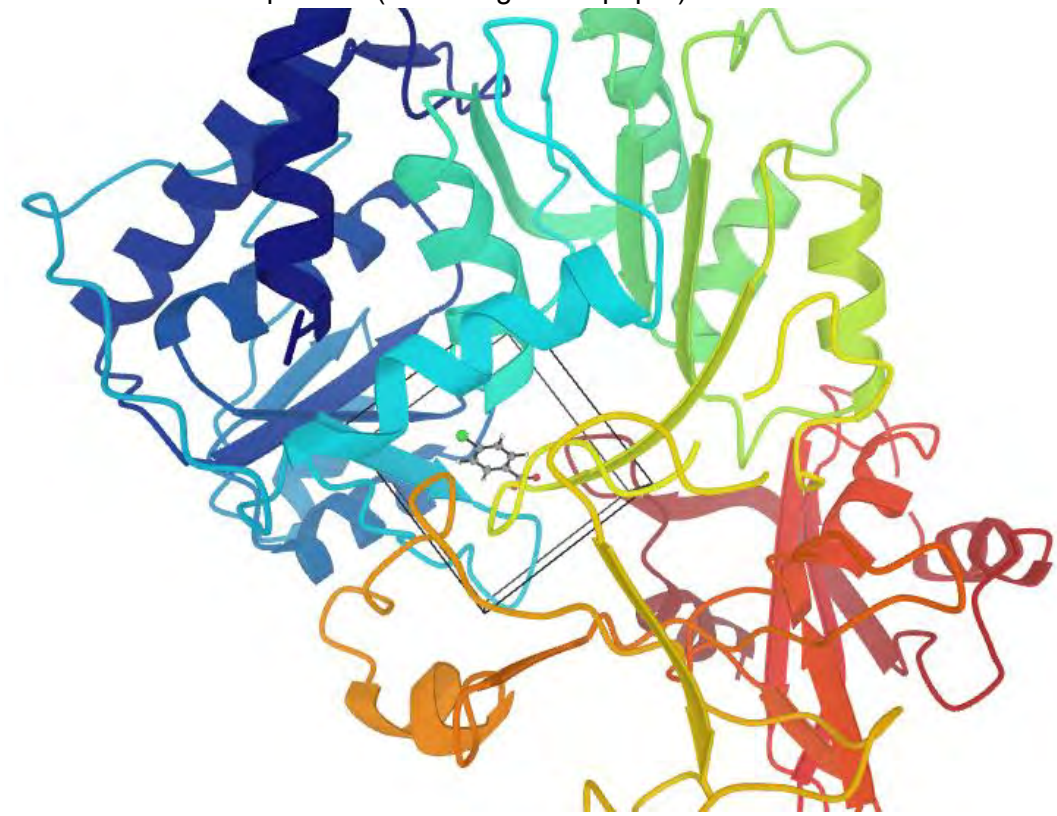
- Seems like they have found a pocket! *Now let's check if they agree with the defined aryl binding pocket in* Crystal structure of 4-chlorobenzoate:CoA ligase/synthetase in the unliganded and aryl substrate-bound states - PubMed:
    - The aryl binding pocket is composed of Phe184, His207, Val208, Val209, Phe249, Ala280, Ile303, Gly305, Met310, and Asn311
- Downloaded PDB file of best binding pose from Rowan… checking for the aryl binding pocket with ChimeraX
- After looking at the residue numbering in 1T5H, it seems that the residue numbers are shifted -5 in the PDB file compared to the paper cited above (I think the paper did not share any PDB files), so that in the file
    - Phe 184 → 179
    - His 207 → 202
    - Val 208 → 203
    - Val 209 → 204
    - Phe 249 → 244
    - Ala 280 → 275
    - Ile 303 → 298
    - Gly 305 → 300
    - Met 310 → 305
    - Asn 311 → 306
- Colored all these residues a different way from the rest of the chain.
- select #1/A:179, 202-204, 244, 275, 298, 300, 305, 306



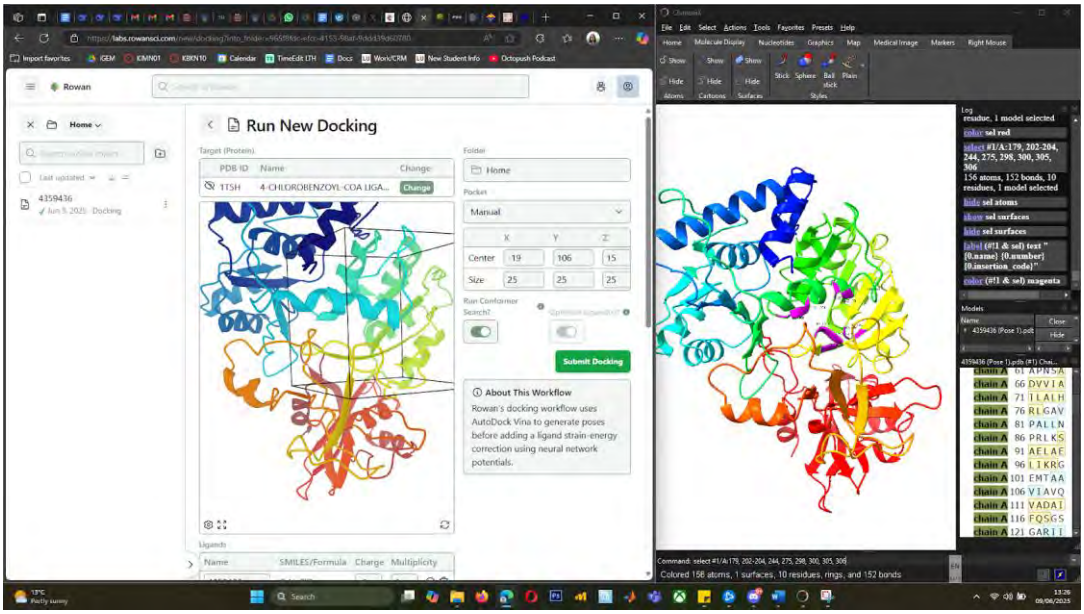
- I think Rowan got the wrong binding pocket! Let's try again….

Attempt #2: Not blind docking

- This time I used the previously docked structure. Let's see if I can move the box to the "correct" position (according to the paper).



- Really hard to match the binding pocket. Is there a way to extract coordinates of residues…



- Starting with a really big box here. See defined binding pocket in magenta on ChimeraX.
- Submitted docking.

- Now it is outside the protein….
- I think we need to find a way to define the binding pocket nicely, but this system works to generate docked PDBs.

Code

No scripts used. Everything was performed using Rowan GUI.

Results Summary

1. Sanitised the structure of 1T5H using the built-in options in Rowan.
2. Performed blind docking of 4-Chlorobenzoate on 1T5H.
3. Docking was completed, and downloaded the results as a PDB file.
4. Checked the docking results using ChimeraX, but the location of the docked ligand is not on the aryl binding pocket mentioned in literature (https://doi.org/10.1021/bi049384m). In the figure below ligand in red and aryl binding pocket residues (Phe184, His207, Val208, Val209, Phe249, Ala280, Ile303, Gly305, Met310, Asn311) highlighted in green.
5. Repeated the docking with defining a box that includes the aryl binding pocket:
6. Docking succeeded, but the ligand is outside of the protein.

The docking results were “successful”, but none resulted in the ligand being docked in the correct aryl binding pocket in literature.

Interpretation

The failure in docking might be related to the fact that 1T5H didn’t have the ligand in the crystal structure. It might be that the unbound enzyme adopts a different conformation from the enzyme bound to the substrate.

Next Steps

Try again with a different crystal structure (different PDB entry). A protein-ligand complex crystal structure would be preferable rather than protein only. We will try again the docking with 1T5D, a crustal structure of 4-Chlorobenzoyl CoA ligase bound to 4-Chlorobenzoate.

2025-06-10 Docking 4-coumarate to 4-coumarate CoA ligase with PyRx

Contributors: Sixten

Objective

Exploring methods for docking and compare with literature to check whether the method is successful.

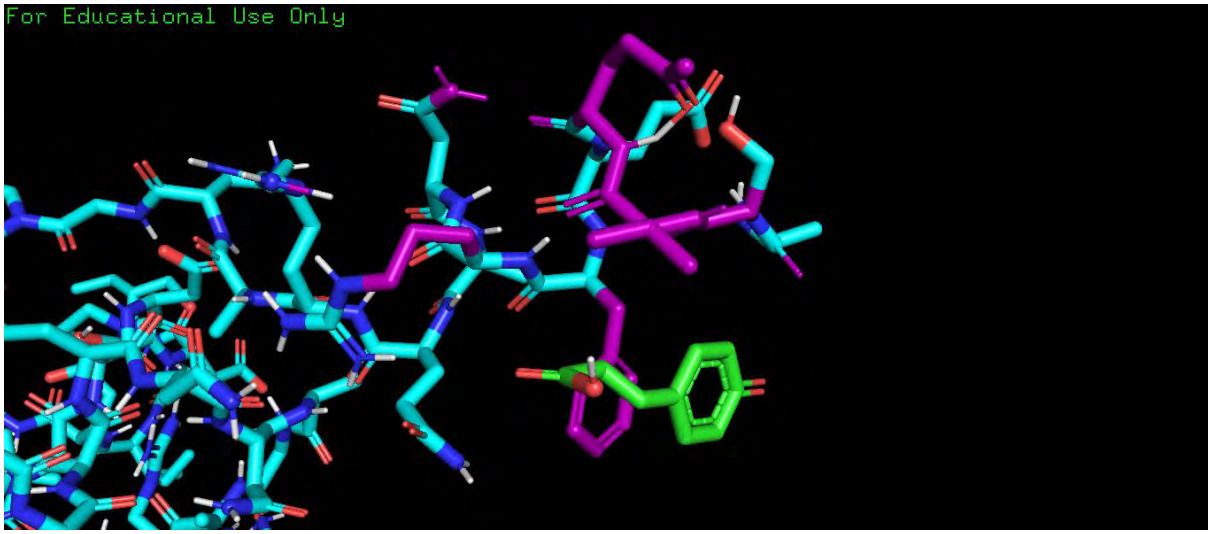
Background

Article covering the binding site of 4-coumarate ligase
https://doi.org/10.1073/pnas.1430550100

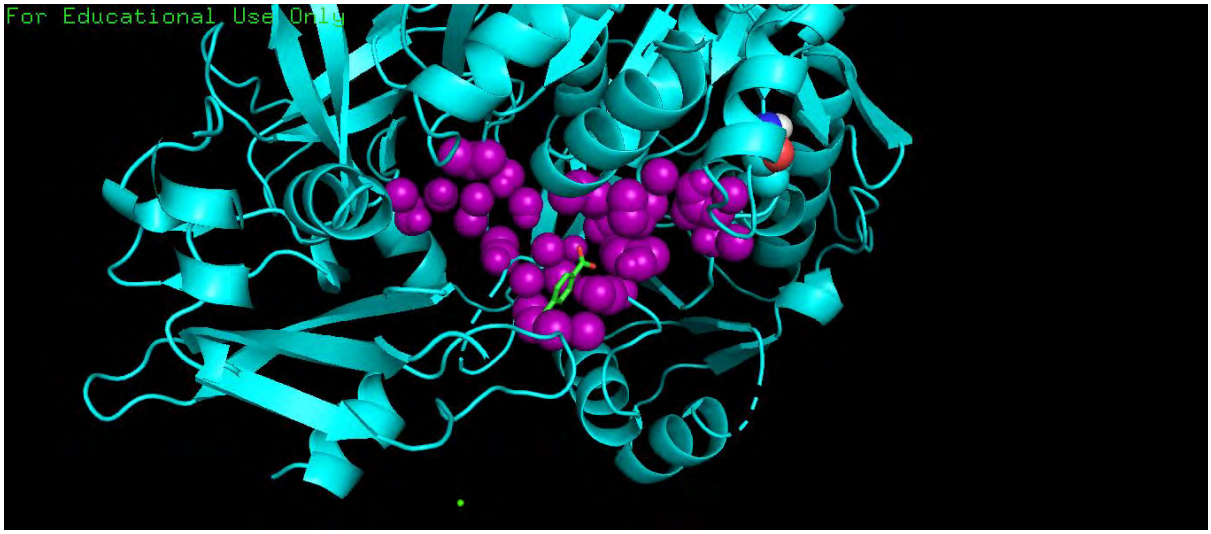
Methods and parameters

Component	*Description*
Software Used	PyRx
Input Structure	3TSY Ligand: 4-coumarate
Key Parameters	Docking score
Script Location	None

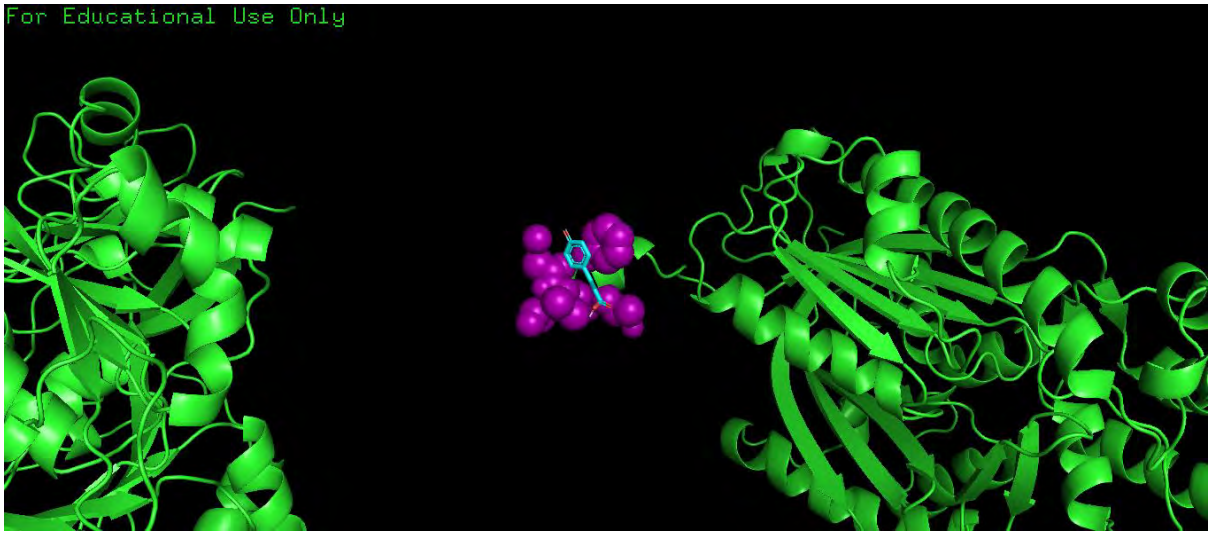
- Downloaded both pdb for 3TSY (4-coumorate CoA ligase) and the sdf for 4-coumarate from uniprot and pubchem respectively.
- Ran PyRx docking
- Coloured the binding site (residues within 4.0 Å) in pymol (purple residues are the ones that are proposed to be involved in binding):



- Feels like the binding site is weird but dono.
- Doing the same with 4-chlorobenzoate ligase



- Remade the image of 4-cholorbenzoate ligase

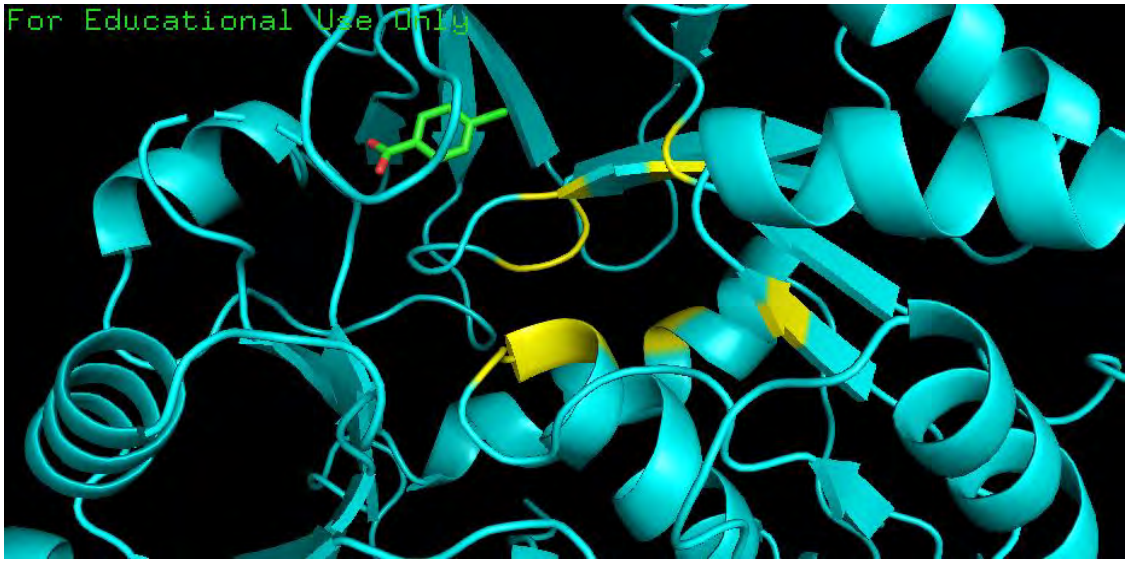


- Downloaded the binding residues as pdb files for each protein respectively. An idea is to compare these residues to the conserved ones using homology search.

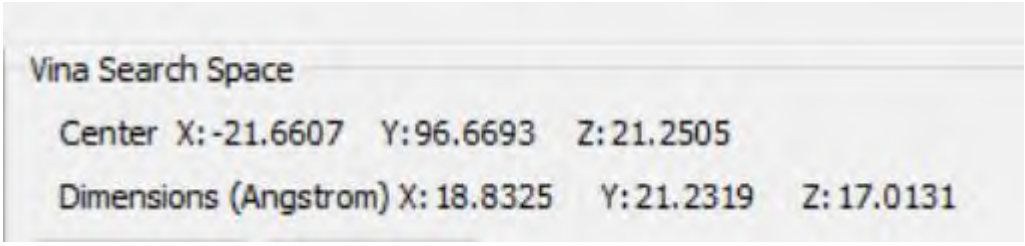
- Trying to move the binding box to get a better docking by finding the coordinates of the residues in the pdb file.
- I do not think the residue numbers are shifted from the paper(?)
- Writing the more extreme coordinates (of each residue, note that the coordinates does not refer to the same atom)

residue (nr)	x	y	z
Phe (184)	-20.493	103.093	16.777
His (207)	-23.070	89.351	17
Val (208)	-26	92	16
Val (209)	-21	92	16
Phe (249)	-15	94	16
Ala (280)	-15	91	22
Ile (303)	-17	97	22
Gly (305)	-21	94	23
Met (310)	-25	97	21
Asn (311)	-22	100	21

- Basing the box on -26, 103 and 23
- For first run the chosen coordinates for center of box are -25,3651; 103,7350; 23,4682
- Got different residues within 4.0 Å, but still not the same as paper. Using the command select binding_site, (receptor within 4.0 Å of ligand). But suspecting there is something wrong with this. It looks weird.
- Docked molecule below and expected binding pocket in yellow, got the same view as Hilya to compare. It seems like we got slightly different results but in the same direction relative to the binding pocket. Binding pocket is highlighted a bit differently due to the -5 residue of Hilya's PDB files that are not necessary in mine. This we have to investigate further.



- Docking was done with



Code

Did everything PyRx, no scripts.

Results Summary

- Ran several tries with varying results but none that seem very good. I will need to change strategy I think.
- Slightly different results from Hilya but in the same direction relative to the binding pocket

Interpretation

Docking still not correct.

Next Steps

- Comparing with Hilya is good.
- Investigate why there is a -5 shift in Hilya's experiments.

2025-06-11 Notes on docking and PyRx

Contributors: Sixten

Objective

Learn more about docking with PyRx and decide what to do next. Come up with a method/pipeline to go from docking to visualisation.

Overview/Background

- Tutorial video explaining docking in PyRx: https://www.youtube.com/watch?v=ld1aF5bBNPU
- It should be possible to use automated scripts to work with pyMOL for visualisation, but have had problems with vscode scripts into pyMOL all day

Results/Discussion

- Since I am waiting for a dry lab meeting I am testing docking with 4-chlorobenzoate-AMP (did not manage to do this).
- Talked with Edu. He thinks the pocket should be open and ready for 4-chlorobenzoate without the ATP binding first. He proposed playing around with the forcefield and trying to make the search box smaller.
- Realized that the reason for the weird binding is that the mechanism involves both ATP and CoA and that these might affect the binding energies. Since the docking is done without these molecules it might work poorly.

Next Steps

- Explore how to use monte carlo-method (have to look into this, a suggestion from Måns)
- Put effort into making python scripts for pyMOL to simplify the docking process.

2025-06-12 Notes on visualisation script for pyMOL

Contributors: Sixten

Objective

Have a better workflow for protein and ligand visualisation, and also the binding site.

Overview/Background

Some references on a potential structure of interest. PDB and the binding site.

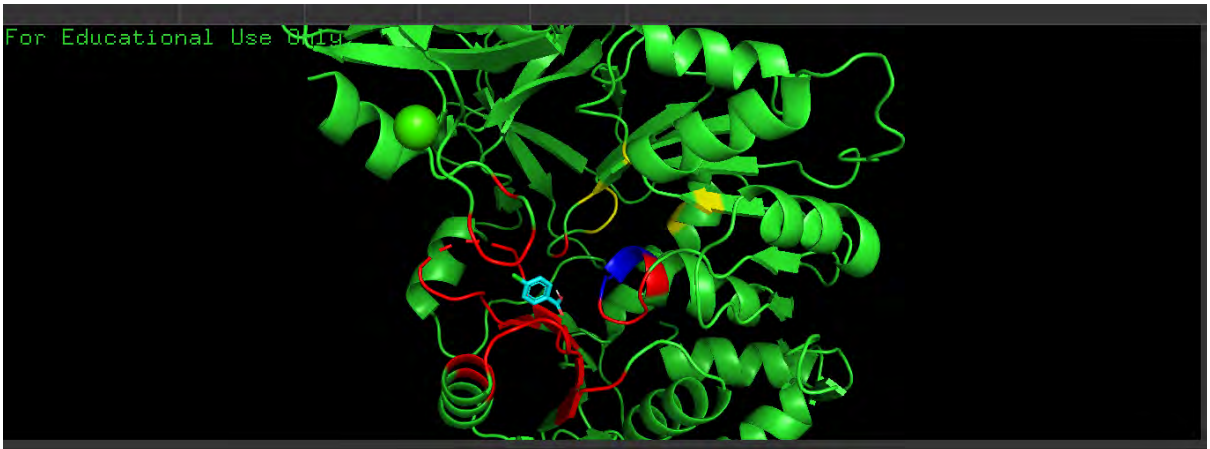
Crystal Structure of 4-Chlorobenzoate:CoA Ligase/Synthetase in the Unliganded and Aryl Substrate-Bound States

Gulick, A.M., Lu, X., Dunaway-Mariano, D.
(2004) Biochemistry **43**: 8670-8679

PubMed: 15236575 Search on PubMed
DOI: https://doi.org/10.1021/bi049384m

Results/Discussion

- Made a script specific for 1t5h that highlights known binding site (from paper), highlights proposed binding site based on distance from ligand, and also compares the two sites to see if any of the residues are the same. The matching residues are colored blue.




```
Known binding site is yellow
Propsed binding site is red
Matching residues are blue
Matching residues = [('X', '207', 'HIS'), ('X', '208', 'VAL'), ('X', '209', 'VAL')]
PyMOL>reinitialize
 CmdLoad: "" loaded as "highexhaust1_receptor".
 CmdLoad: "" loaded as "highexhaust1_ligand".
PyMOL>z
PyMOL>run 1t5h.py
Known binding site is yellow
Propsed binding site is red
Matching residues are blue
Matching residues = [('X', '207', 'HIS'), ('X', '208', 'VAL'), ('X', '209', 'VAL')]
```

- Things to note are that the script only works if receptors and ligands end with _receptor or _ligand respectively. This is however easy to change. Another thing to note is that the script only accepts one receptor and one ligand. This can be modified so that it can handle several ligands.
- Link to script: Binding site comparison

```
#Description:
#Highlights the selection of resides that is your active site
#Colours the residues close to your ligand red
#Compares the residues close to your ligand with the chosen active site and
colours matching residues blue
#If you have loaded a selection of several ligands and named them "..._ligand"
#the script checks which of the ligands fit the binding site the best

from pymol import cmd

# Gets object names
objects = cmd.get_names()

# Selects known binding site from paper
kbs = cmd.select("known_binding_site", "resi
184+207+208+209+249+280+303+305+310+311")

# Colors the binding site yellow
cmd.color("yellow", "known_binding_site")
```

```
ligands    = [obj for obj in objects if obj.endswith('_ligand')]

# Gets residues for known binding site
residues_kbs = set()
cmd.iterate('known_binding_site', "residues.add((chain, resi, resn))",
space={"residues": residues_kbs})

# Track best match
best_match_count = 0
best_ligand_indices = []
matching_data = {}

# Loop over ligands to determine proposed binding sites
for i, ligand in enumerate(ligands):
    pbs_name = f'proposed_binding_site{i}'
    cmd.select(pbs_name, f'({receptors[0]} within 5.0 of {ligand})')

    residues_pbs = set()
    cmd.iterate(pbs_name, "residues.add((chain, resi, resn))",
space={"residues": residues_pbs})

    matching_residues = [res for res in residues_pbs if res in residues_kbs]
    match_count = len(matching_residues)
    matching_data[i] = {
        "count": match_count,
        "residues": matching_residues,
        "resi_numbers": [res[1] for res in matching_residues]
    }

    # Update best match info
    if match_count > best_match_count:
        best_match_count = match_count
        best_ligand_indices = [i]
    elif match_count == best_match_count:
        best_ligand_indices.append(i)

# Handle coloring and output
for i in range(len(ligands)):
    cmd.color('red', f'proposed_binding_site{i}')  # All proposed binding
sites are red
```

```
print('Known binding site is yellow')
print('Proposed binding sites are red')

if best_match_count == 0:
    print('There are no matching residues')
elif len(best_ligand_indices) == 1:
    i = best_ligand_indices[0]
    matching_resi = "+".join(matching_data[i]["resi_numbers"])
    cmd.select(f'matching_residues{i}', f'resi {matching_resi}')
    cmd.color("blue", f'matching_residues{i}')

    final_matching_residues = set()
    cmd.iterate(f'matching_residues{i}', "residues.add((chain, resi, resn))",
space={"residues": final_matching_residues})

    print('Matching residues are blue')
    print(f'Matching residues = {final_matching_residues}')
    print(f'The ligand with the highest number of matching residues is:
{ligands[i]}')
else:
    print(f'Multiple ligands have the same number of matching residues
({best_match_count})')
    print('Ligands with highest match count:')
    for i in best_ligand_indices:
        print(f'- {ligands[i]}')
```

Next Steps

- Would be nice to make a script that takes several ligand positions from a docking, compares the proposed binding sites with the known one and then highlights the ligand position with the most matching residues.

2025-16-16 Docking 4-chlorobenzoate to 1T5H using PyRx

Contributors: Sixten

Objective

Successful docking of 4-chlorobenzoate to the correct binding site of 1T5H.

Background

Some previous docking attempts has failed to put the ligand in the correct binding site described in the literature. These include blind dockings or dockings performed with a defined box. Today we want to try again with a more defined (smaller) box.

Methods and parameters

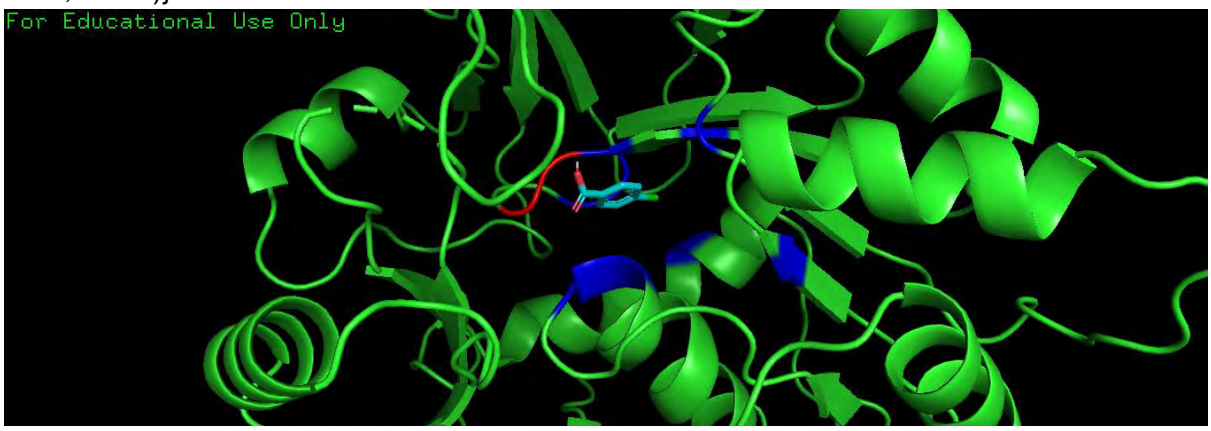
Component	*Description*
Software Used	PyRx
Input Structure	1T5H
Key Parameters	Binding affinity, RMSD
Script Location	Extract box

- Trying to dock with a smaller box
- Following coordinates were used

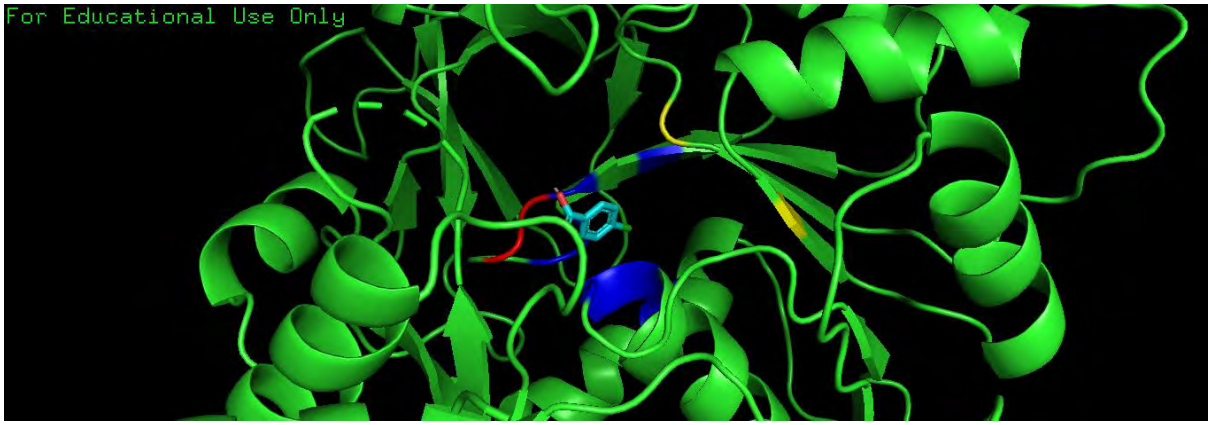
- Results inside of the binding pocket! The blue residues are the residues within 5 and 4 Å respectively from the ligand that are matching the proposed binding pocket.

Within 5.0 Å:
Known binding site is yellow
Proposed binding site is red
Matching residues are blue

Matching residues = {('X', '311', 'ASN'), ('X', '303', 'ILE'), ('X', '305', 'GLY'), ('X', '184', 'PHE'), ('X', '208', 'VAL'), ('X', '310', 'MSE'), ('X', '209', 'VAL'), ('X', '207', 'HIS'), ('X', '280', 'ALA'), ('X', '249', 'PHE')}



Within 4.0 Å:
Known binding site is yellow
Proposed binding site is red
Matching residues are blue
Matching residues = {('X', '311', 'ASN'), ('X', '303', 'ILE'), ('X', '305', 'GLY'), ('X', '184', 'PHE'), ('X', '208', 'VAL'), ('X', '310', 'MSE'), ('X', '209', 'VAL'), ('X', '207', 'HIS')}



- But… the binding energy is VERY bad. These are the results from PyRx:

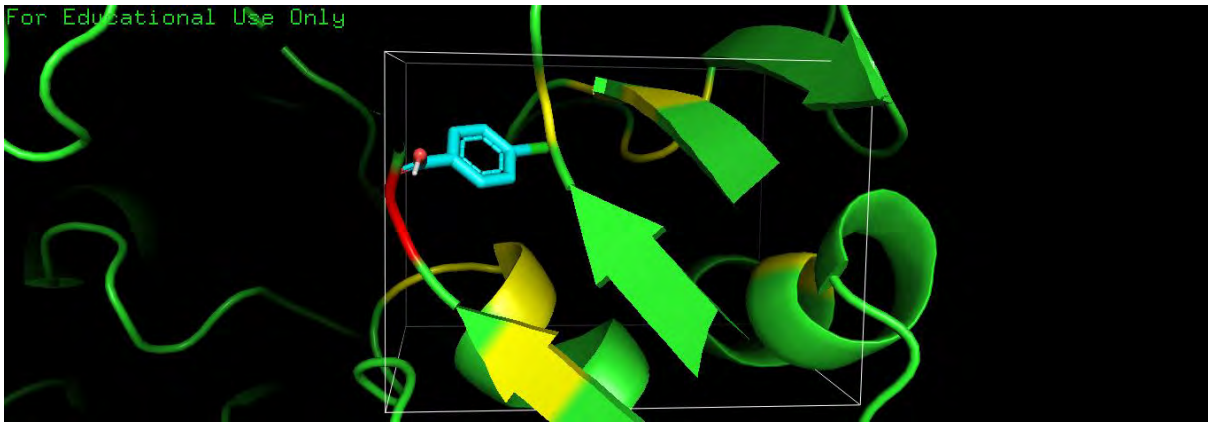
Name of ligand	binding affinity (kcal/mol)	RMSD (lower bound)	RMSD (upper bound)
pdb1t5h_4-chlorobenzoate_uff_E=56.64	-0.2	10,124	11,169

- A good binding affinity is (-5) - (-6) kcal/mol and -4 kcal/mol is accepted. -0.2 kcal/mol is really low.
- I think maybe some residues where not counted, so making the box a bit bigger

New box:

```
Vina Search Space
  Center  X: -20.5260 Y: 94.7920  Z: 22.3309
  Dimensions (Angstrom) X: 16.9773    Y: 17.3532    Z: 10.6383
```

- Name of files are 1small_box
- Results were similar but a little worse
- Realized you can extract box-coordinates from pymol. Let chatGPT make the script.
- Seems to work. This is output:




```
Bounding Box:
  Min coords: [-26.062999725341797, 86.28800201416016, 11.118000030517578]
  Max coords: [-11.20300006866455, 103.09300231933594, 23.323999404907227]

Use these for docking:
  Center     = [-18.632999897003174, 94.69050216674805, 17.220999717712402]
  Box size   = [14.859999656677246, 16.80500030517578, 12.205999374389648]
```

Code

Used visualisation scripts for PyMOL.

```python
#Description:
#Extracts the coordinates of a box around a selection of residues

from pymol import cmd
from pymol.cgo import BEGIN, LINES, COLOR, VERTEX, END

def draw_bounding_box(selection="known_binding_site"):
    # Step 1: Get extent (bounding box corners)
    min_coords, max_coords = cmd.get_extent(selection)

    print("Bounding Box:")
    print(f"  Min coords: {min_coords}")
    print(f"  Max coords: {max_coords}")
```

```
    # Step 2: Compute center and size
    center = [(min_coords[i] + max_coords[i]) / 2.0 for i in range(3)]
    size = [max_coords[i] - min_coords[i] for i in range(3)]

    print("\nUse these for docking:")
    print(f"  Center     = {center}")
    print(f"  Box size   = {size}")

    # Step 3: Draw box using CGO
    x0, y0, z0 = min_coords
    x1, y1, z1 = max_coords

    edges = [
        (x0, y0, z0, x1, y0, z0),
        (x0, y0, z0, x0, y1, z0),
        (x0, y0, z0, x0, y0, z1),
        (x1, y1, z1, x0, y1, z1),
        (x1, y1, z1, x1, y0, z1),
        (x1, y1, z1, x1, y1, z0),
        (x0, y1, z0, x1, y1, z0),
        (x0, y1, z0, x0, y1, z1),
        (x0, y0, z1, x1, y0, z1),
        (x0, y0, z1, x0, y1, z1),
        (x1, y0, z0, x1, y1, z0),
        (x1, y0, z0, x1, y0, z1),
    ]

    box_obj = [BEGIN, LINES, COLOR, 1.0, 1.0, 1.0]  # White box
    for e in edges:
        box_obj.extend([VERTEX, *e[:3], VERTEX, *e[3:]])
    box_obj.append(END)

    cmd.load_cgo(box_obj, "bounding_box")

# Run the function on 'known_binding_site'
draw_bounding_box()
```

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

Successfully docked 4-chlorobenzoate into the correct binding site described in literature. Results:

Name of ligand	binding affinity (kcal/mol)	RMSD (lower bound)	RMSD (upper bound)
pdb1t5h_4-chlorobenzoate_uff_E=56.64	-0.2	10,124	11,169

Interpretation

- Binding energy is VERY bad
    - A good binding affinity is (-5) - (-6) kcal/mol and -4 kcal/mol is accepted. -0.2 kcal/mol is really low.

Next Steps

Figure out the reason for the bad binding energy. Try this method with the other substrates and enzymes. Try with TPA.

2025-06-17 Docking 4-chlorobenzoate and 4-chlorobenzoate AMP

Contributors: Sixten

Objective

Explore docking in PyRx using different ligands, 4-chlorobenzoate-AMP vs 4-chlorobenzoate. See if using a smaller defined box for ligand position will help with docking.

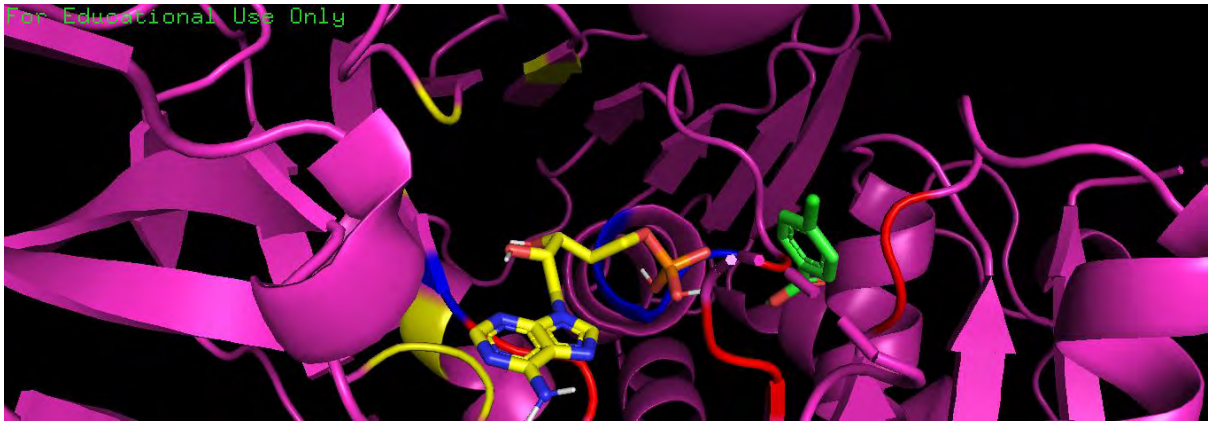
Background

We previously performed docking of 4-chlorobenzoate but with bad results. Want to try with different structures, substrate states, box sizes, etc.

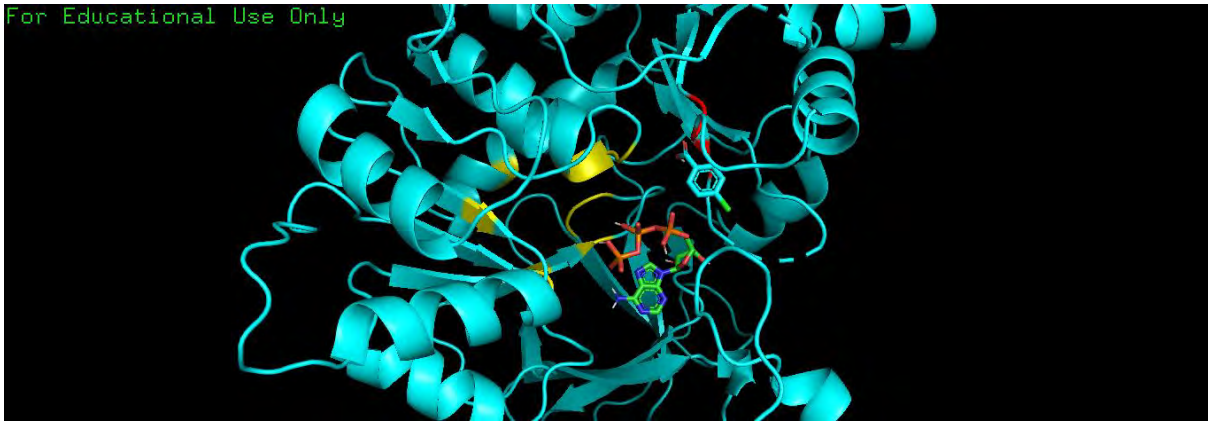
Methods and parameters

Component	*Description*
Software Used	PyRx, pyMOL
Input Structure	Ligands: 4-Chlorobenzoate 4-Chlorobenzoate AMP Protein: 1T5H
Key Parameters	Affinity Position Ligand strain
Script Location	Extract box

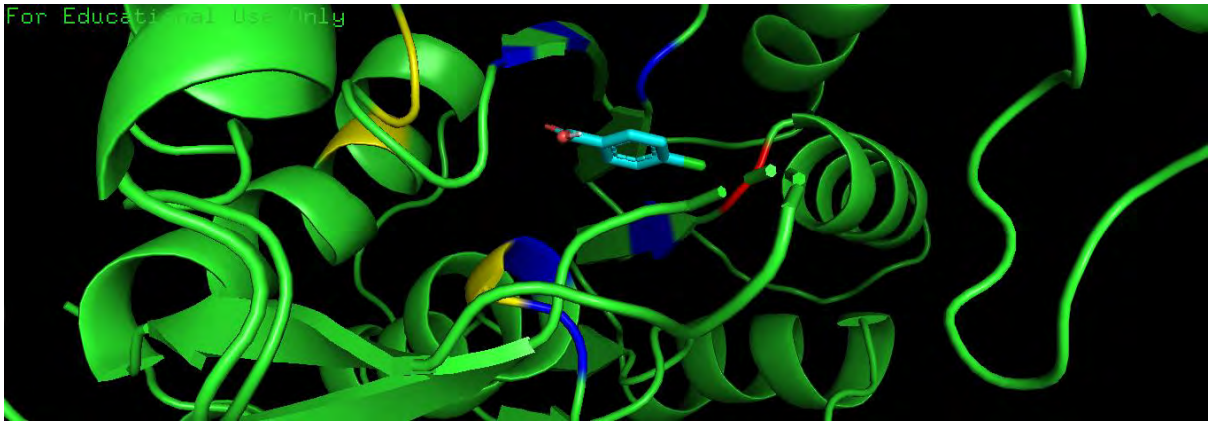
- Running another docking with the given coordinates. Naming the files 2small_box.
- Still not in the binding pocket, but maybe getting closer. Trying one more time by making the box even smaller. The smaller box was made manually based on the script that extracts coordinates for a box around a selection of residues.
- Still cannot get it to fit in the pocket.
- End of ideas. Trying to do multiple ligand docking with AMP
- After docking with AMP, substrate is still not in binding site



- After docking with ATP

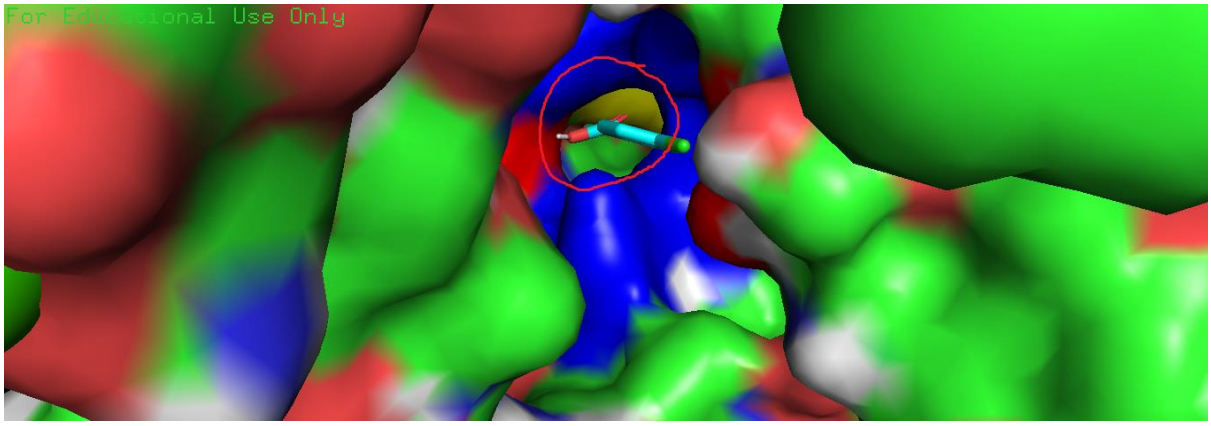


- From ChatGPT: In PyRx, minimizing the ligand's energy optimizes its 3D geometry by refining its bond angles, torsions, and overall conformation to create a more realistic starting structure before docking, which directly impacts how well it will later fit into the target binding site. The choice of force field for this minimization is critical. The Universal Force Field (UFF) is a broad, generic option that covers the entire periodic table, making it acceptable for ligands with inorganic atoms or unusual functional groups, or for quick preliminary screens. However, for most organic, drug-like molecules, the Merck Molecular Force Field (MMFF94/s) is strongly recommended. It is specifically designed for such compounds, providing superior accuracy in modeling torsional strain, aromatic systems, and hydrogen bonding, which is essential for obtaining reliable docking poses and any subsequent analysis.
- This is interesting because UFF was used to minimize energy. Will try using the other forcefield instead (mmff).
- Worked ok. This was the ligand closest to the binding site. Binding affinity = -3,8 kcal/mol (ok)

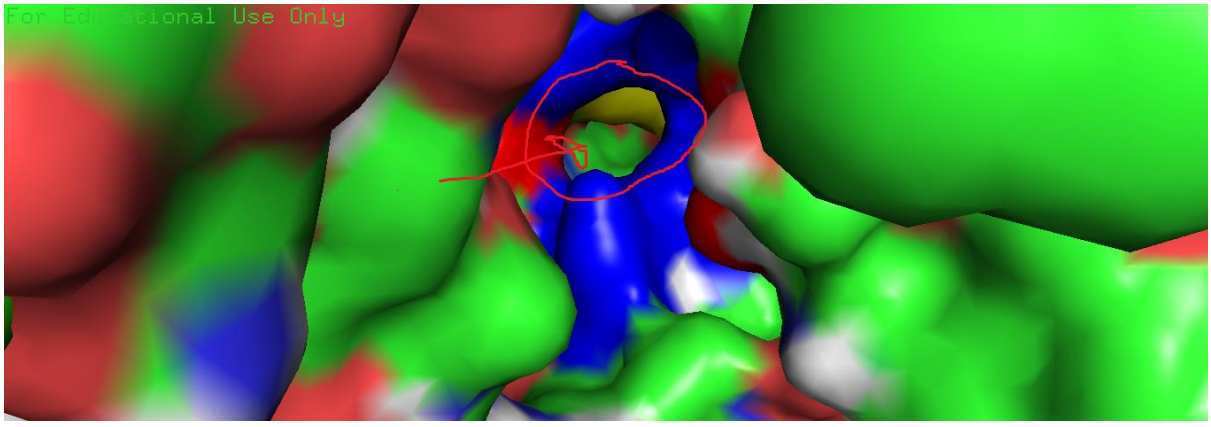


- When looking at the surface of the protein we can see the binding pocket more clearly. It seems the hole that the substrate has to go through is a bit small(?)

With ligand:

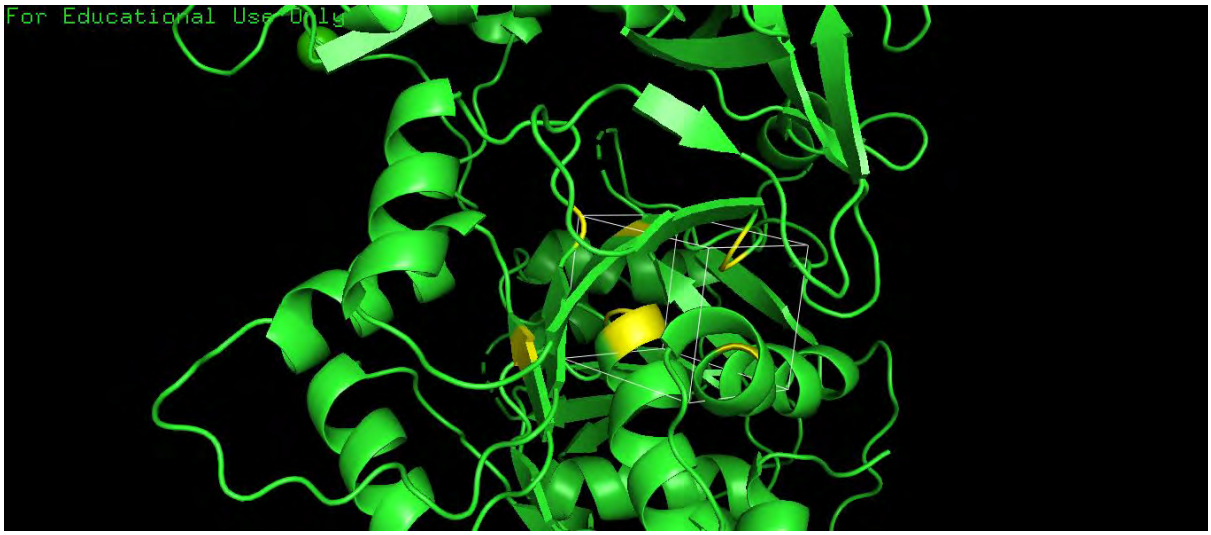


Without ligand:

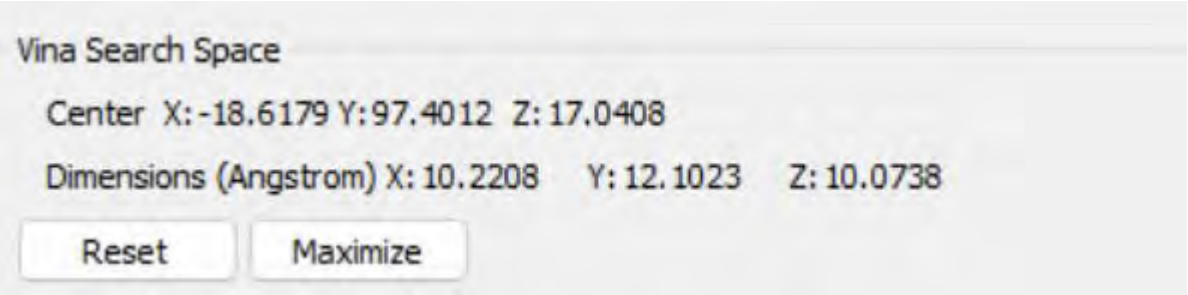


- Binding pocket is on the other side (or inside) of the hole the arrow is pointing at.
- According to chatGPT, the downloaded pdb file for 1t5h is in the intermediate form, i.e the conformation it is in after it has bound AMP-coumaryl. This structure is likely not the exact same as the structure when 4-chlorobenzoate is binding into the pocket. Could this be the reason for the weird docking?

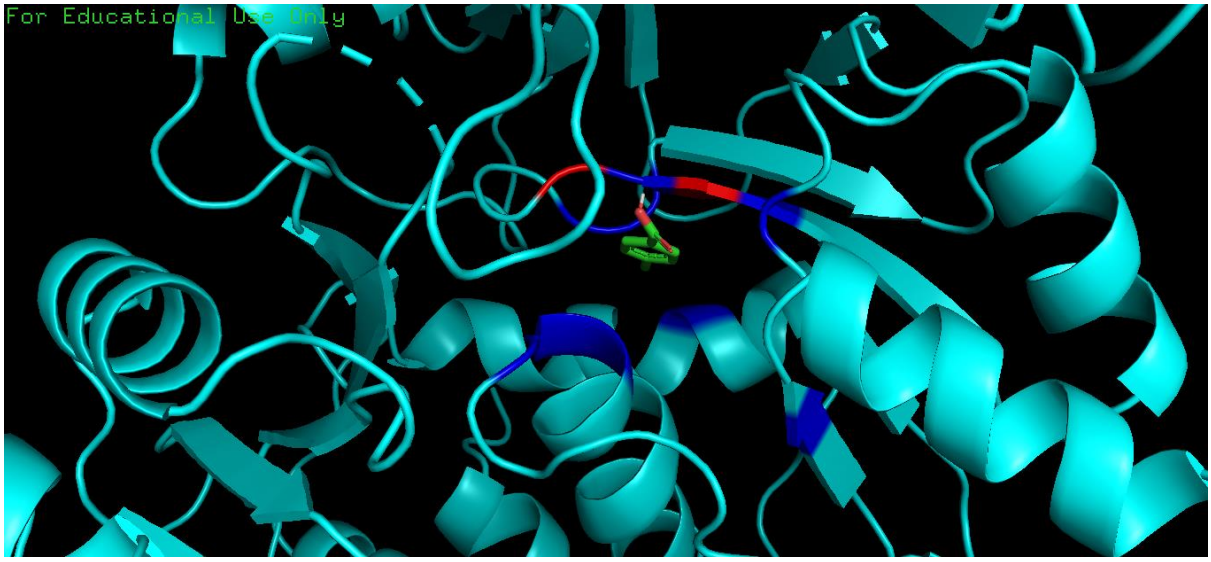
- Right now I would want to explore the idea of docking the 4-chlorobenzoate-AMP intermediate.
- Downloaded the 4-chlorobenzoate-AMP sdf-file from pubchem, converted to pdb, minimized the energy in PyRx using the mmff-method and converted the file to pdbqt.
- Performing one docking without mg2+, however this should probably be added somehow.
- Did give very high binding affinities (-7,5 to -8,0) but none where the 4-chlorobenzoate part was in the binding pocket. The seventh docked molecule was the best one (in regards to binding pocket).
- To create a macromolecule with the mg2+ ion I need to use Autodock Tools. This requires an updated graphics card with OpenGL to run. Will fix.
- Successfully updated graphics-card and opened autodock tools. Having trouble with understanding the program though. Will try to find a tutorial.
- Going to watch this tutorial: https://www.youtube.com/watch?v=k6tqCeDIwEk
- Remade script for extracting box according to Edu's suggestions. Works great. I am now trying to dock using this box:



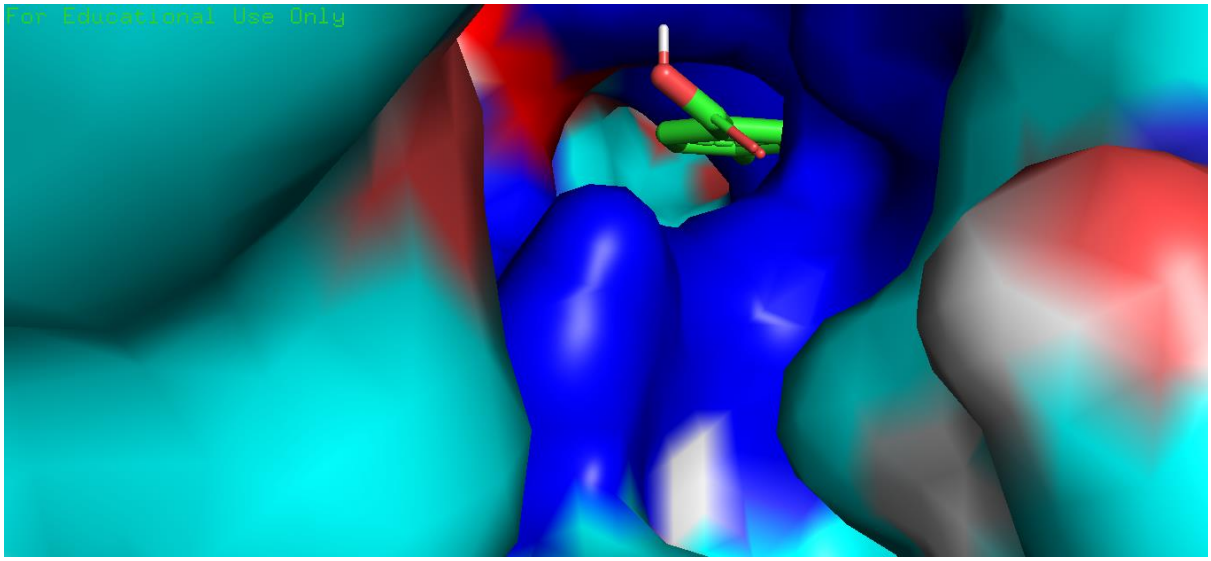
- Coordinates:



- WORKED! Name of the files is 4small_box. All the residues specified in the paper are within 5.0 Å. Crazy. But the docking is VERY not favorable. It is super weird. All of them hade binding affinities > 0 (i.e super not favorable).



Surface:



- ChatGPT writes:

Possible Reasons for a Positive Affinity Score Despite a Good-Looking Pose

- 1. Improper File Preparation:
  - Missing hydrogens, incorrect partial charges, or bad torsion definitions in the receptor or ligand PDBQT files.
  - Incorrect protonation states of key residues or the ligand at physiological pH.
- 2. Steric Clashes:
  - The pose may have subtle but energetically costly atomic overlaps with the protein's sidechains or backbone, even if it appears to fit visually.
- 3. Incorrect Docking Box Size:
  - A box that is too small can artificially strain the ligand during docking, forcing it into a high-energy conformation.
- 4. Limitations of the Scoring Function:
  - Scoring functions (like Vina's) are approximations and often fail to account for:
    - Solvent effects and entropy.

- Metal coordination chemistry.
- Full protein flexibility.

- They may misinterpret the energy of certain interactions.

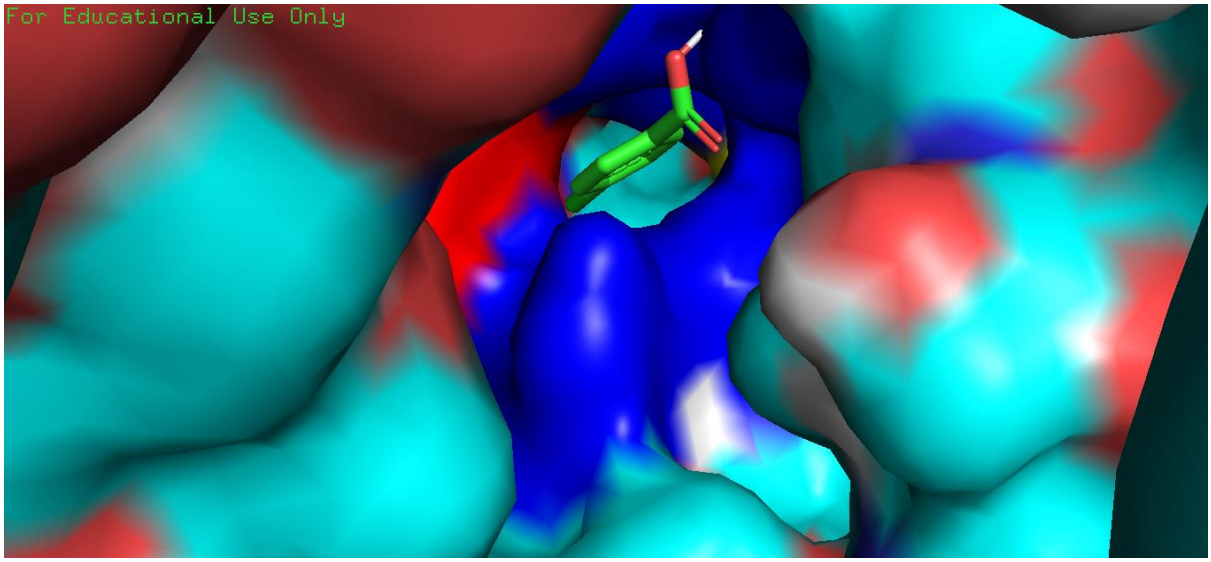
5. The Ligand Itself May Be Inactive:
   - The molecule may lack key chemical features for strong binding (e.g., hydrophobic anchors, specific H-bond donors/acceptors), meaning it fits the shape but not the chemistry of the pocket.

Recommended Troubleshooting Actions

Action	Purpose
Re-check PDBQT file generation	Ensure proper protonation, charges, and rotatable bond definitions.
Visually inspect for steric clashes	Identify atomic overlaps that the scoring function penalizes.
Increase the docking box size	Remove artificial strain by giving the ligand more space to maneuver.
Dock a known positive control	Validate that your setup can produce the expected negative ΔG for a good binder.
Compare poses with a co-crystal ligand	Identify any missing key interactions in your docked pose.
Try a different tool or re-score with a more advanced method (e.g., MM-GBSA)	Rule out limitations of the Vina scoring function specifically.

Summary: A positive ΔG with a good pose often points to a technical issue in preparation or a scoring function limitation. Always suspect preparation errors first before concluding the ligand is bad.

- This is how the ligand with the best binding affinity (1,8 kcal/mol) was oriented:



Results Summary

Using a smaller box helps dock the ligand in the right place. Initial dockings of did give very high binding affinities (-7,5 to -8,0) but none where the 4-chlorobenzoate part was in the binding pocket. The seventh docked molecule was the best one (in regards to binding pocket), but with bad binding energy (delta is positive).

Interpretation

The enzymes are in different shapes according to which substrate/intermediate they bind.

Next Steps

Always match protein structure with the ligand state.

2025-06-18 Identifying the active site of 4-chlorobenzoyl CoA ligase: docking 4-chlorobenzoate to 1T5D

Contributors: Sixten

Objective

Validating Rowan docking results by comparing a docked structure and the crystal structure with the bound ligand.

Background

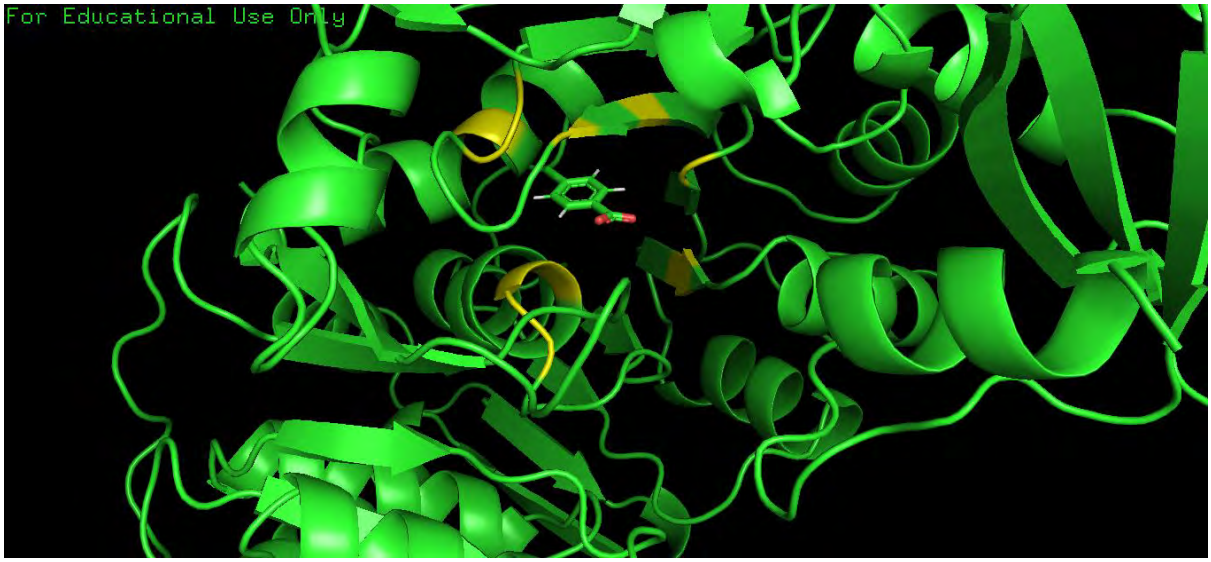
During meeting Hilya did docking in Rowan again but with the 1T5D structure, which already had a 4-chlorobenzoate docked. Previously the structure we used is 1T5H which doesn't have the ligand, which might be incompatible. She got it to work with good results (good binding affinity). I tried the same in PyRx but it did not work, so it seems there is something wrong with my docking protocol for PyRx. Right now, future docking will be done in Rowan if this is successful.

Methods and parameters

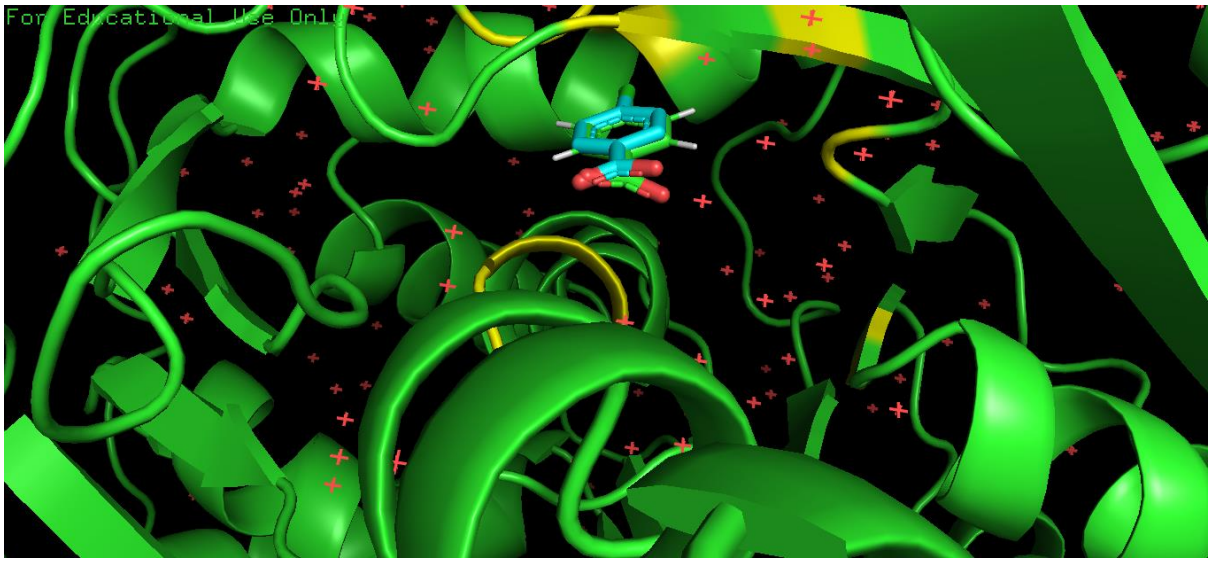
Component	*Description*
Software Used	Rowan, pyMOL
Input Structure	Protein: 1T5D Ligand: 4-Chlorobenzoate
Key Parameters	Docking score
Script Location	None

- Downloaded Rowan and trying to get the same results.

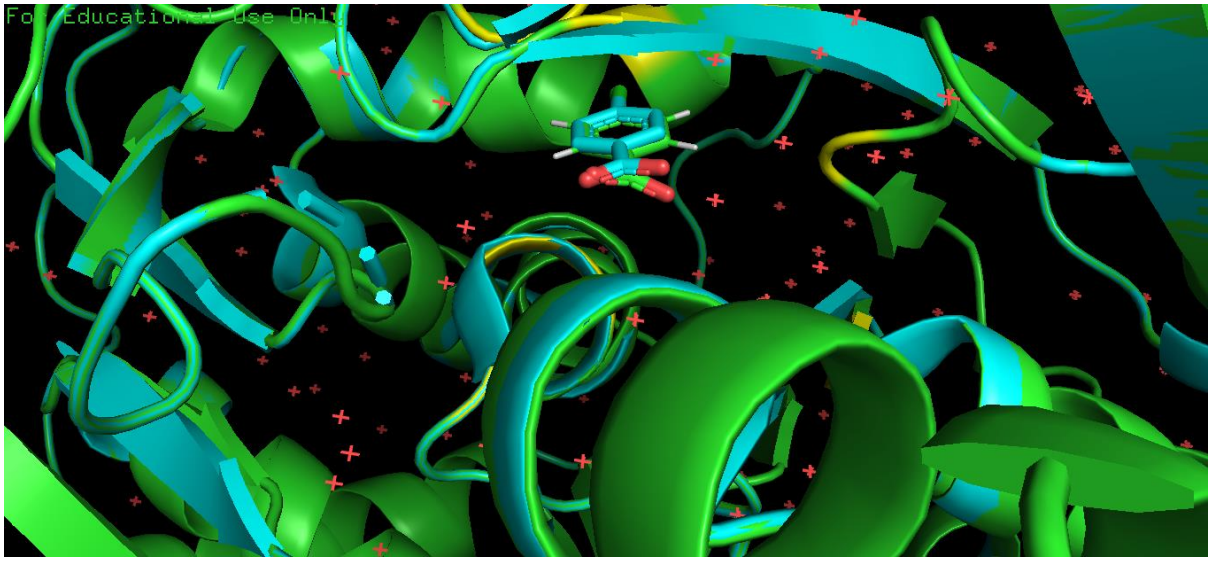
Docked PDB:



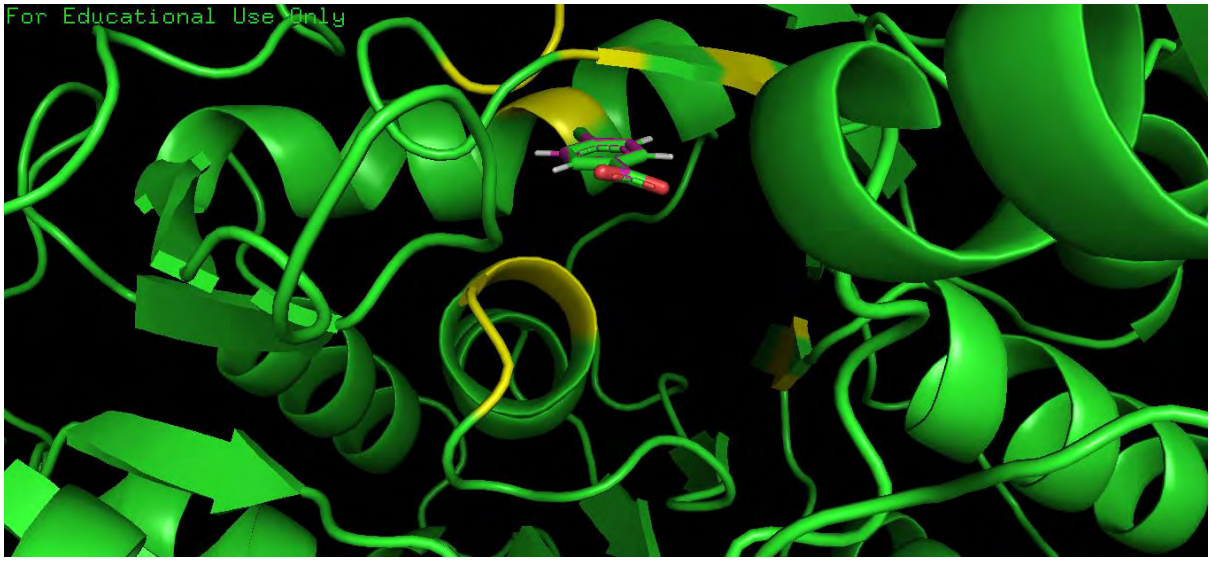
Docked pdb compare with the downloaded docked ligand:



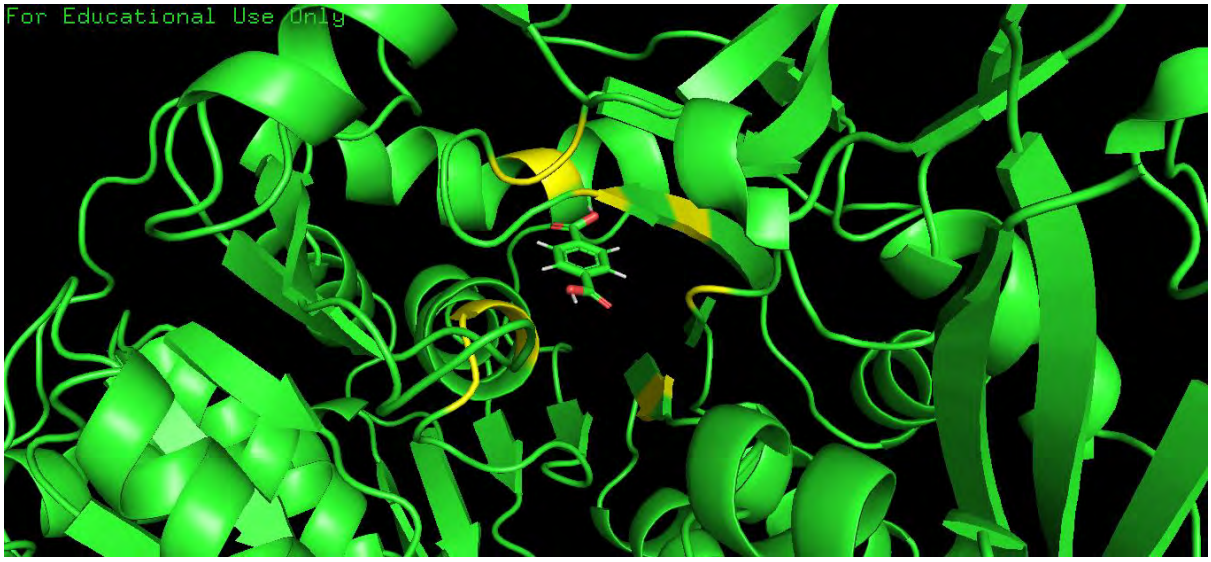
Docked pdb compared with the docked structure:



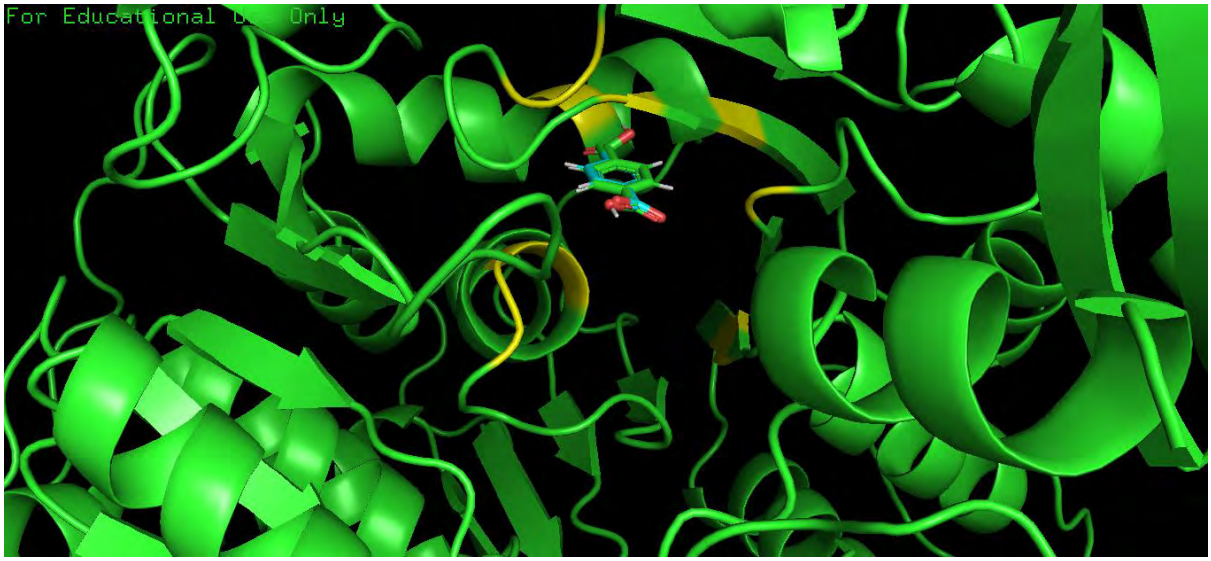
Docked pdb compared with Hilya's docking (appears to be exactly the same):



- Conclusion: use Rowan for future docking.
- Performing docking with TPA to see results. Realized when doing this that the carboxylic group on the 4-chlorobenzate is charged while the carboxylic group on the TPA molecule is not (it has its hydrogens). Is this relevant? At physiological pH the carboxylic acids should be charged I think.
- The docking was fairly successfull!



Compared with docking of 4-chlorobenzoate (4-chlorobenzoate is the blue one):



- Investigating steric hindrance of the docked TPA molecule

Code

None.

Results Summary

- Worked! Got the same results as Hilya!

Interpretation

1T5H might be incompatible with the ligand and not good to be used as a structure, might be an inactive conformation.

Next Steps

- Realized when doing this that the carboxylic group on the 4-chlorobenzate is charged while the carboxylic group on the TPA molecule is not (it has its hydrogens). Is this relevant? At physiological pH the carboxylic acids should be charged I think.
    - Try again with charged TPA.

2025-06-24 Comparing and investigating characteristics of aryl substrates

Contributors: Sixten

Objective

Comparing and investigating the characteristics of TPA, 4-chlorobenzoate, and other ligands to explore mutation strategies for 4-chlorobenzoate ligase. Visualise differences in how they sit in the active site of the enzyme.

Background

Understanding the properties of the substrate will hopefully give us an idea on how they interact with enzymes. With Rowan, it appears that you can create a PCA plot that gives a visual representation of how similar molecules are. I will download more molecules from the aryl-binding paper to compare with TPA, visualise the steric differences and hopefully come up with ideas for mutation.

Methods and parameters

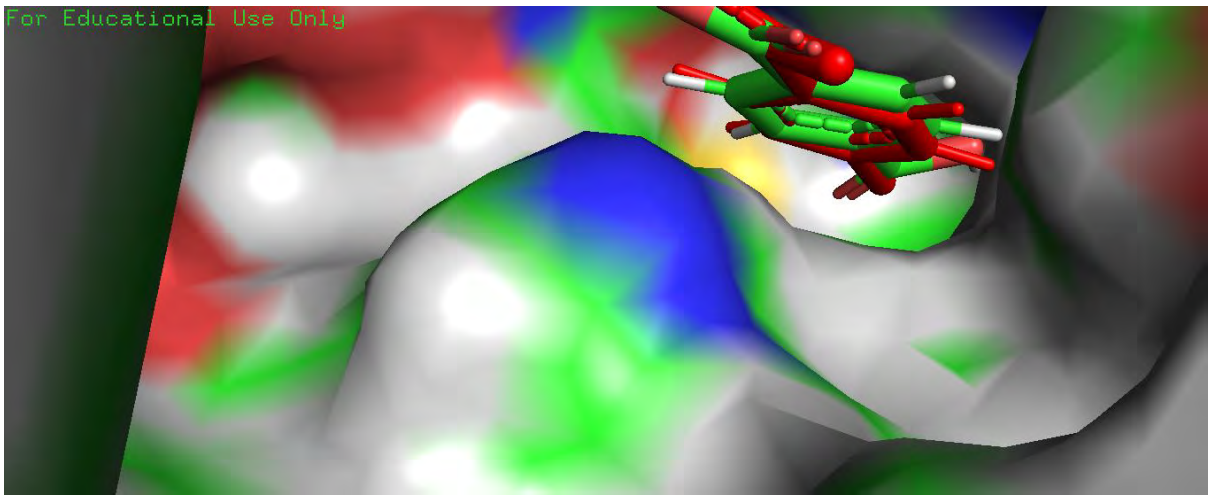
Component	*Description*
Software Used	Rowan, PyMOL
Input Structure	Protein structure: 1T5D. Downloaded molecules are: Benzoate, 2-fluorobenzoate, 3-fluorobenzoate, 2-aminobenzoate, 4-bromobenzoate, 4-iodobenzoate, 4-methylbenzoate, 4-hydroxybenzoic acid, cyclohex-1,4-dienecarboxylate, 3-hydroxybenzoic acid, 4-aminobenzoate, 4-hydroxybenzoic acid, Phenylactetate, TPA and 4-fluorobenzoate.
Key Parameters	Steric clashes
Script Location	No scripts used.

Doing this by using the descriptor calculation in Rowan.

Method: the video tutorial on Rowans website regarding this
https://docs.rowansci.com/tutorials/submit/descriptors

- Had an error when running the PCA in Rowan, but messaged someone working there and they would look at it. Waiting for an answer.
- Did a comparison in pymol of the distance to residues to TPA and 4-chlorobenzoate respectively to see if there was a difference. It seems the residue ASN 311 is within 3,0 Å of the TPA molecule but not within 3,0 Å of the 4-chlorobenzoate, suggesting potential steric clashes. Is it possible to change a ligand in the binding site? Or what is the idea here? If we can change a ligand in the binding site then a smaller residue than ASN could be sufficient.
- RMSD, Binding energy, Make a negative control by making a non conservative change.
- Dock 4-chlorobenzoate to the mutated protein as well.

Here is a picture of the surface of the docked TPA (green ligand). It can be seen that there are probably some steric clashes present.



Code

No scripts used.

Results Summary

Since the PCA had an error no conclusion was drawn.

Conclusion on the visualization is that the binding pocket is very small for the ligand, not giving a lot of room for the TPA

Interpretation

The residue ASN 311 is within 3,0 Å of the TPA molecule but not within 3,0 Å of the 4-chlorobenzoate, suggesting potential steric clashes. Is it possible to change a ligand in the binding site? Or what is the idea here? If we can change a ligand in the binding site then a smaller residue than ASN could be sufficient.

Next Steps

Contact the experts about why the docking might work for 1T5H and not for 1T5D.

Questions for meeting: Does the charged TPA/non-charged TPA matter when doing docking? Now that we have successfully docked TPA, how do we proceed to do the point mutations? Silvia mentioned that they did this in the biophysical I think. Is it the same workflow?

Had an error when running the PCA in Rowan, but messaged someone working there and they would look at it. Waiting for an answer.

2025-06-25 Docking uncharged TPA vs doubly-charged (deprotonated) TPA to 1T5D with Rowan

Contributors: Sixten

Objective

Evaluate the difference between docking a doubly charged/deprotonated form of TPA and protonated TPA to enzyme. See if it was the reason behind failed dockings.

Background

From Terephthalic Acid | C8H6O4 | CID 7489 - PubChem it seems that TPA should be doubly charged form in physiological pH. So we should dock that instead of fully protonated TPA.

pKa Type
pKa
Temperature [°C]
Citation
pKa1
3.5
20
M. L. Dondon, J. Chim. Phys. Phys-Chim. Biol. 54, 304 (1957).
pKa2
4.34
20
A. V. Willi and J. F. Stocker, Helv. Chim. Acta 38, 1279 (1955).
pKaH1
-7.51
not_stated
R. Stewart and M. R. Granger. Can. J. Chem. 39, 3508 (1961).

Methods and parameters

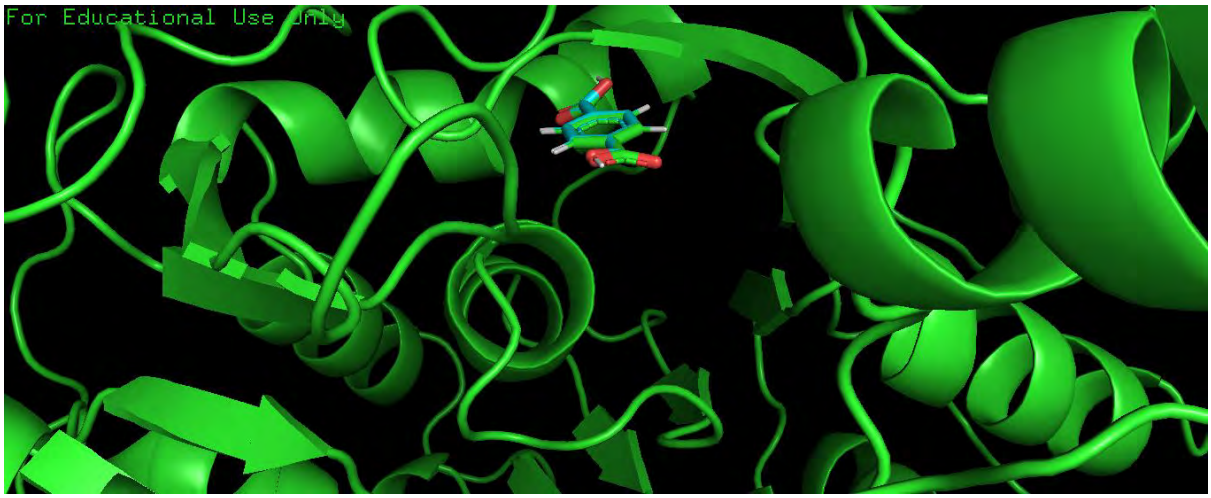
Component	*Description*
Software Used	
Input Structure	Ligands: TPA, doubly-charged.
Key Parameters	
Script Location	

- Realized you can edit molecules in Rowan directly! What an amazing website.

- Changing TPA so that it is in its charged form and performing the docking.
- Super interesting results from the docking with the charged TPA. Binding affinity stayed the same but the strain on the molecule is significantly better. Previous docking had 1,355.

Pose	Docking Score	Ligand Strain (kcal/mol)
1	-2.9...	0.174
2	-2.8...	7.231

Charged vs uncharged docking. It can be concluded that there is practically no difference between the position of the docked ligand



- Regarding emailing the authors of the paper investigating the 4-chlorobenzoate ligase it seems the article is from 2004. Maybe too long time ago?

Code

No scripts used.

Results Summary

Not very much position difference between docking the charged and uncharged structures, but significantly better strain.

Interpretation

Charge definitely has a role in binding, so we should use the correct form when performing dockings or cofoldings.

Next Steps

Always dock with doubly charged TPA.

2025-06-27 Docking of TPA to mutants of 1T5D

Contributors: Sixten, Hilya

Objective

Compare how 4-chlorobenzoate (4CB) and terephthalic acid binds to 4-chlorobenzoate ligase and the effect of mutating one of the residues in the active site to how the ligand sits inside the binding pocket.

Background

One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

We want to try a few mutations:

Mutation	Rationale
Asn 311 → Thr	Thr and Ser has similar properties to Asn (polar uncharged), but a smaller. We hope that this can reduce the steric pressure on TPA (which is bigger due on that side to the carboxylic acid).
Asn 311 → Ser	Thr and Ser has similar properties to Asn (polar uncharged), but a smaller. We hope that this can reduce the steric pressure on TPA (which is bigger due on that side to the carboxylic acid).

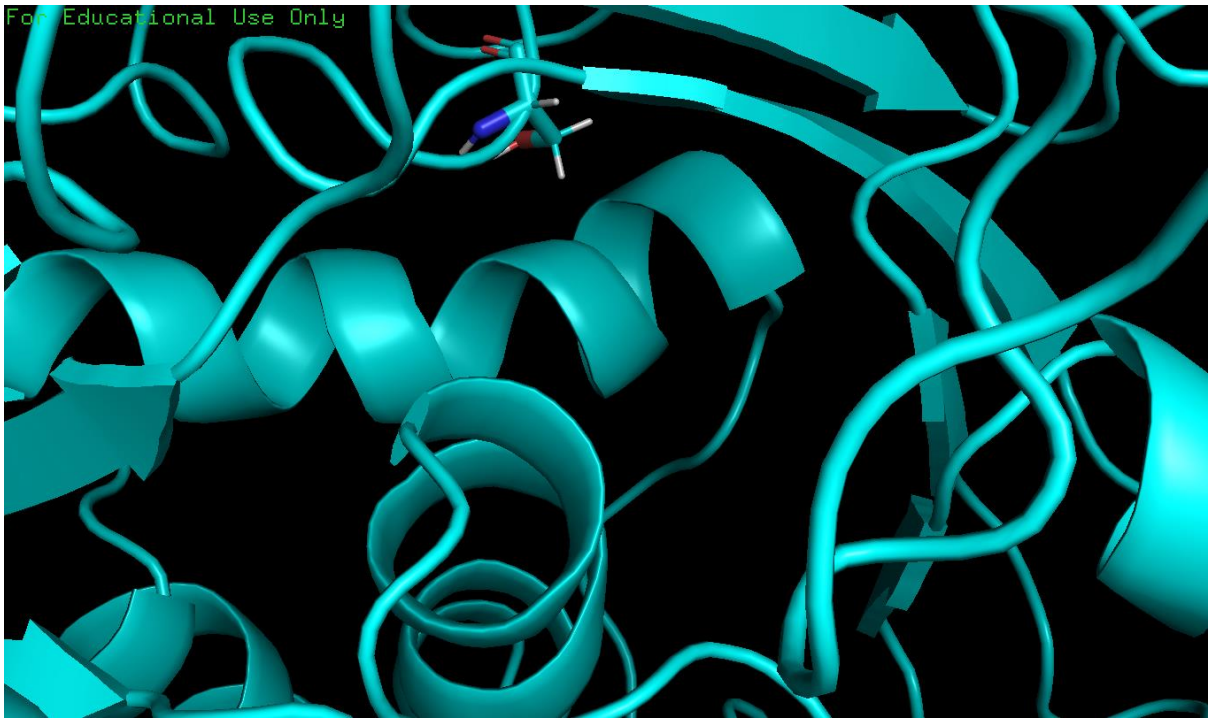
Methods and parameters

Component	Description
Software Used	Rowan, PyMOL
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem

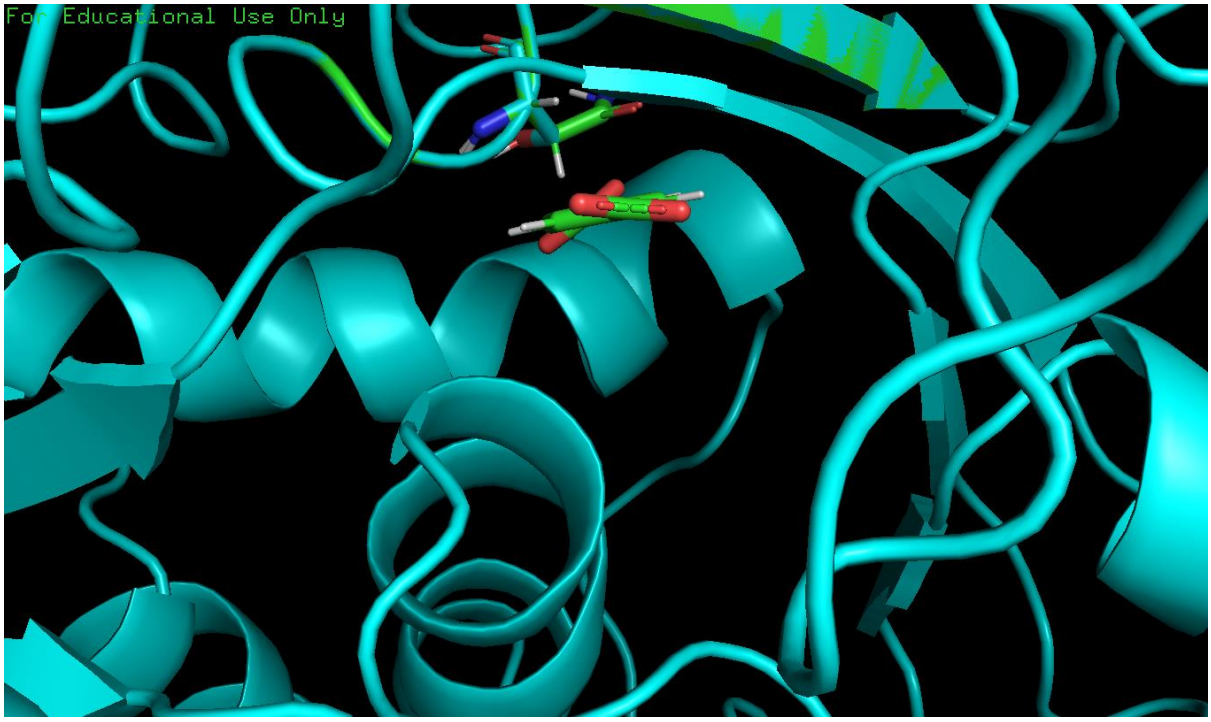
Key Parameters	Steric clashes, visualisation
Script Location	

- Going to decide which point mutations to make. So far the suggestions have to make the 311 residue Asn into a smaller residue with similar properties. Edu suggested threonine or serine. Another difference between 4-chlorobenzoate and the TPA molecule is that is has a charge very close to the
- Performed point mutation in the pyMOL wizard mutagenesis. Changed Asn 311 to a Ser. Did make it smaller:

Mutated Serine (not that the TPA is not present in this pdb when only cartoon is showing. It appears when sticks are showing however, a bit weird):

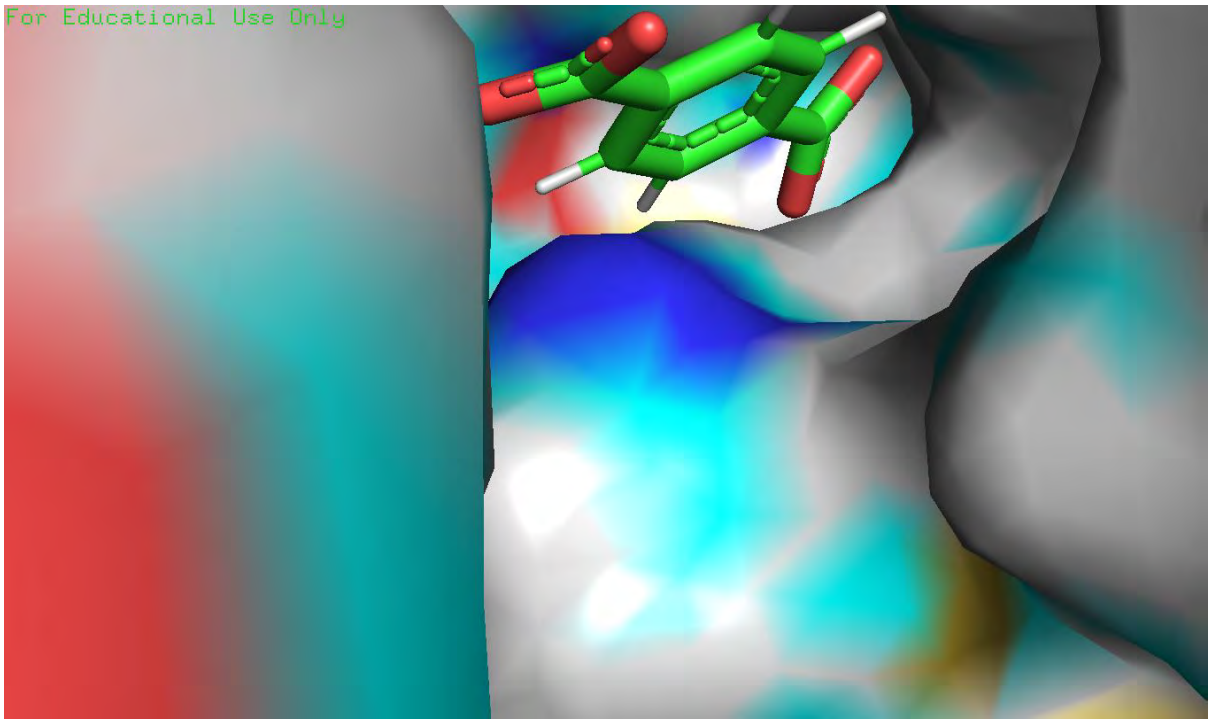


No mutation:

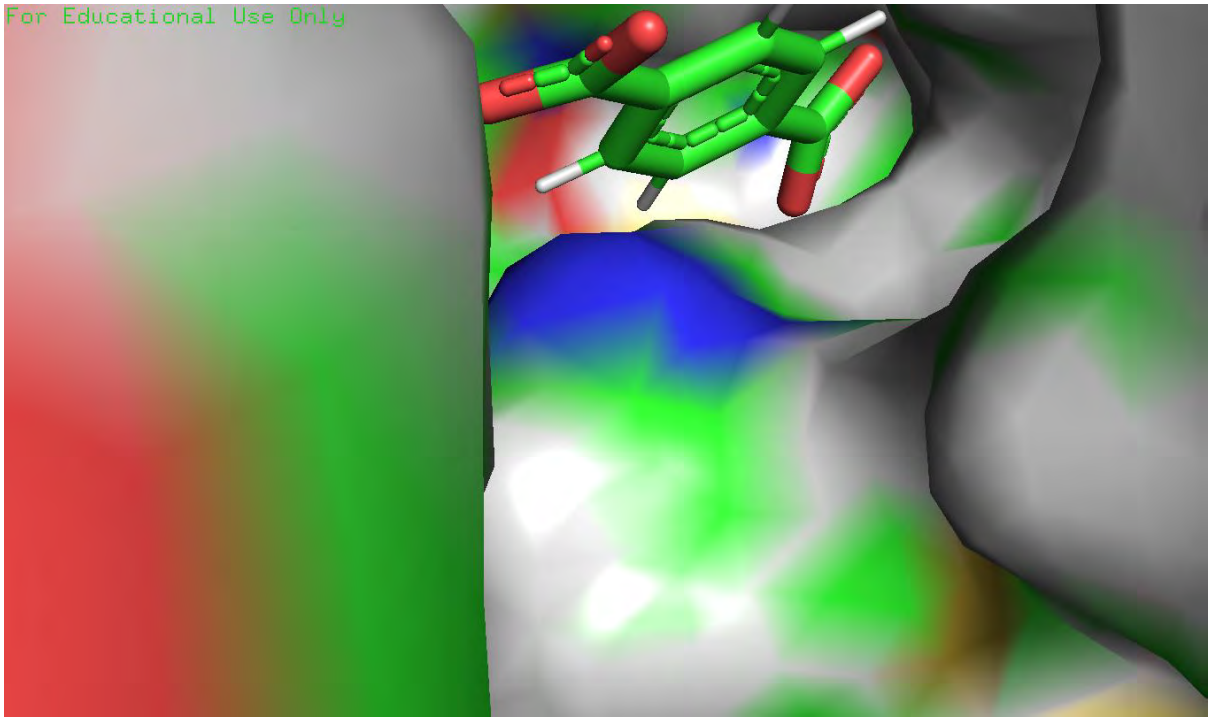


- But when looking at the surface it seems the mutation did nothing or at least very little.

Mutated surface:



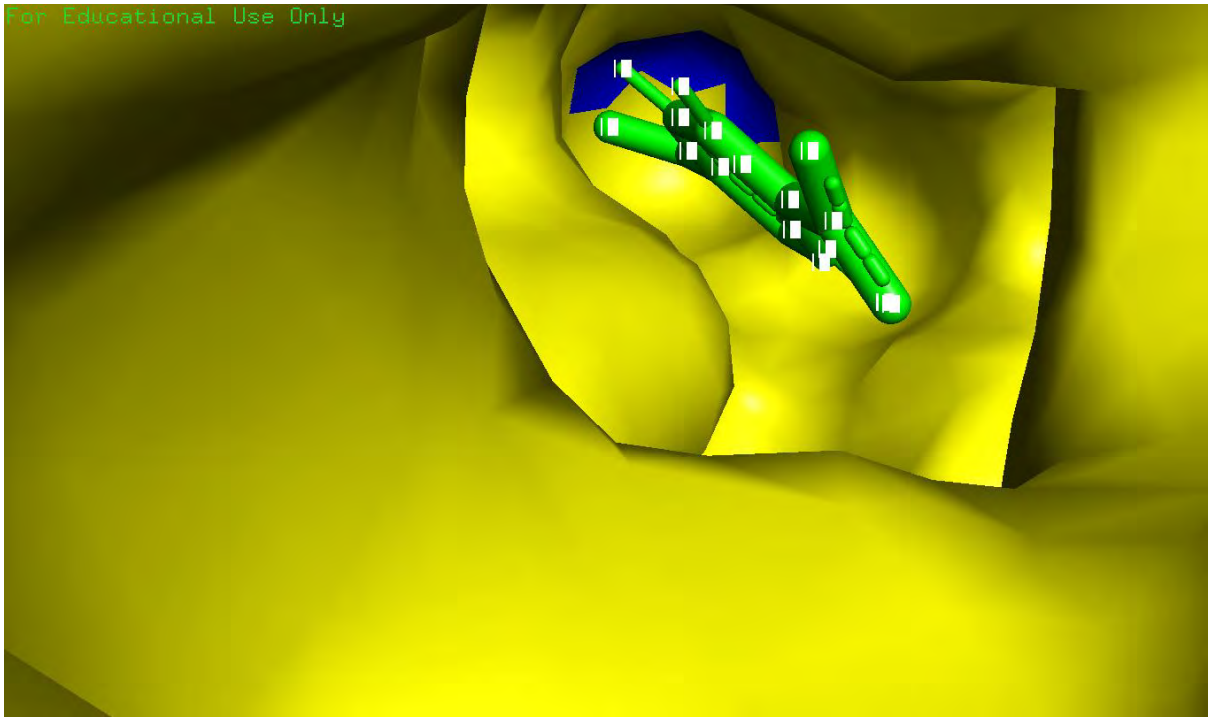
No mutation surface:



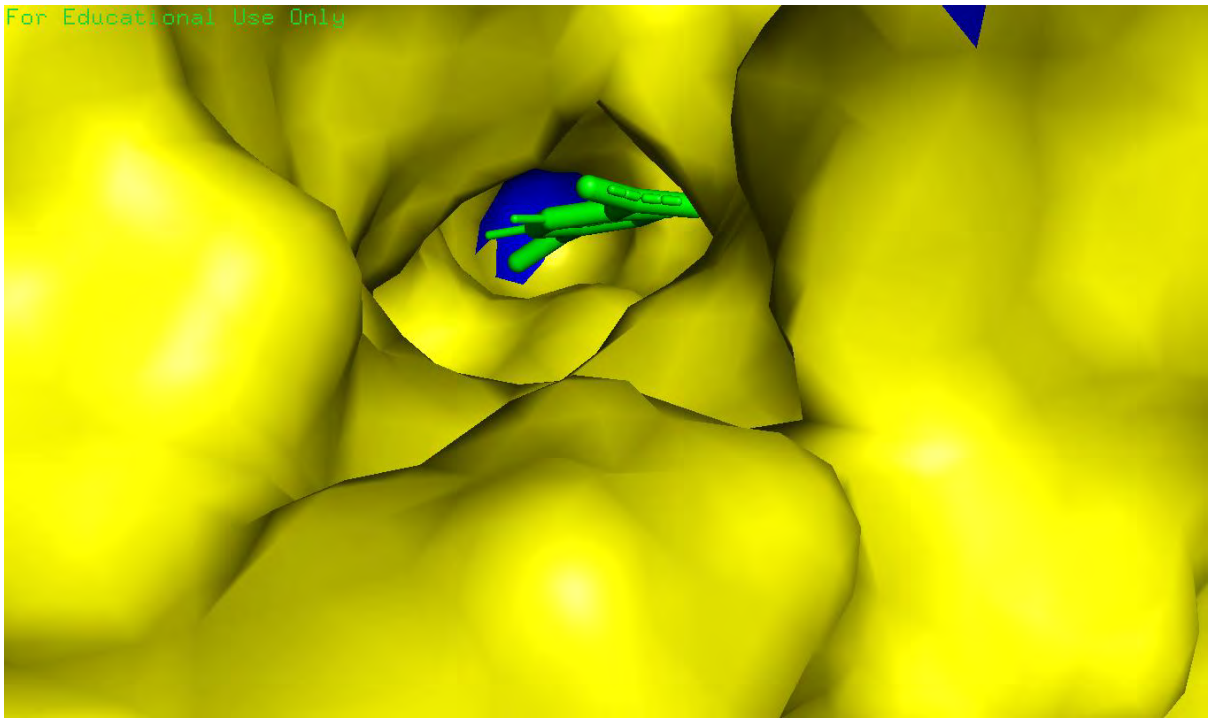
- It appears that there are most likely other residues responsible for the steric clashes.
- Chimera better at camera viewing, tip from Hilya (JUST A MEETING NOTE)
- Overlap surfaces (Should do this for all mutations)
- Download chimera (JUST A MEETING NOTE)
- Also noticed this on Rowan website:

Citing Rowan. To cite calculations from the Rowan platform, please refer to Rowan's documentation. If you publish a paper or preprint citing Rowan, please let us know and we'll be happy to share it with our network!
About AIMNet2 (ωB97M-D3). This calculation employs the AIMNet2 neural network potential developed by Dylan Anstine, Roman Zubatyuk, and Olexandr Isayev at CMU. The PyTorch models can be found on GitHub. If you use these results, please cite the article.

- Maybe we should have a conversation with them about our project. They might be willing to sponsor us with a subscription, other things etc
- Did the docking on the Asn 311 → Ser and it did not give better results. Slightly worse binding affinity of similar position. Also did the control docking with 4-chlorobenzoate to the mutated protein and got similar results, i.e slightly worse binding affinity (-3,1 instead of -3,2) and worse ligand strain.
- Here is an overlap of the surfaces Asn 311 → Ser. Maybe a slight difference but really similar overall. Yellow is the old surface and blue is the mutated surface.



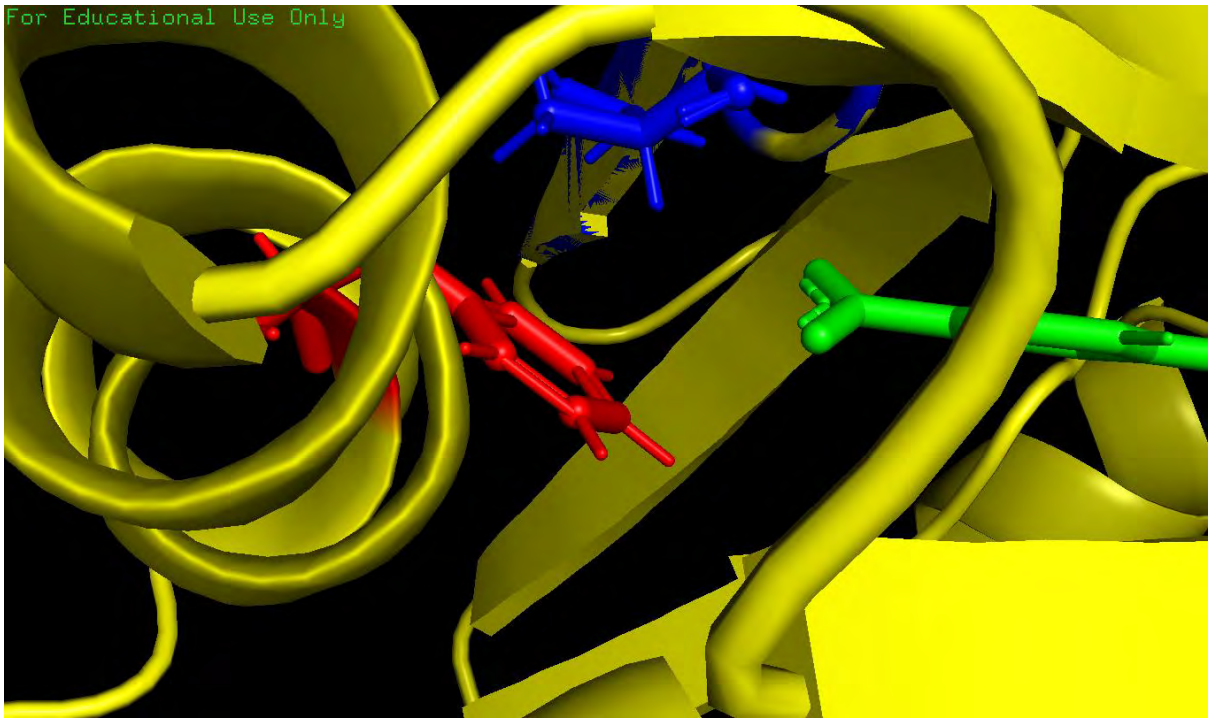
- Trying to make the Asn 311 → Threonine mutation now.
- The overlapping surfaces:



- Results on the Asn 311 → Thr was similar to Asn 311 → Ser. Conclusion: Maybe change which residue is targeted.

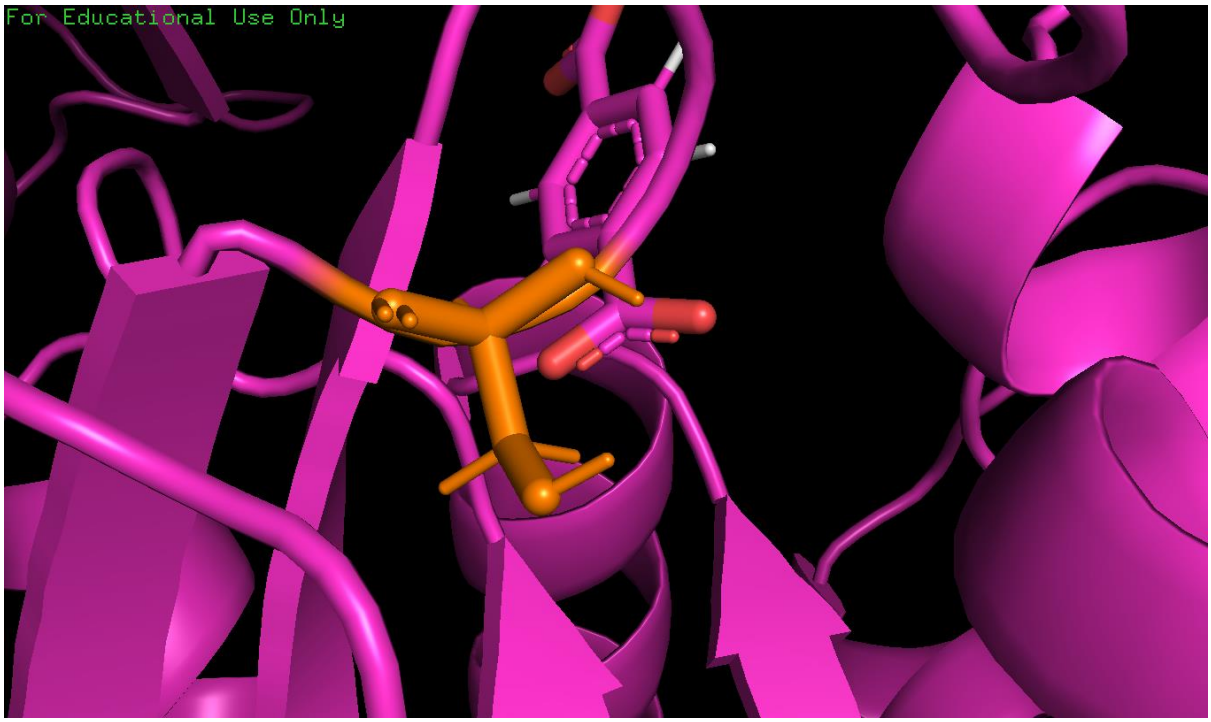
Here is a distance comparison. The blue is the Asn 311 residue, red is the Phe 184 residue and green is charged TPA. Think the Phe might be sterically hindering the TPA. But it is part

of secondary structure, do not know if this is relevant when doing point mutations as changing the residue here might affect the structure of the protein significantly.

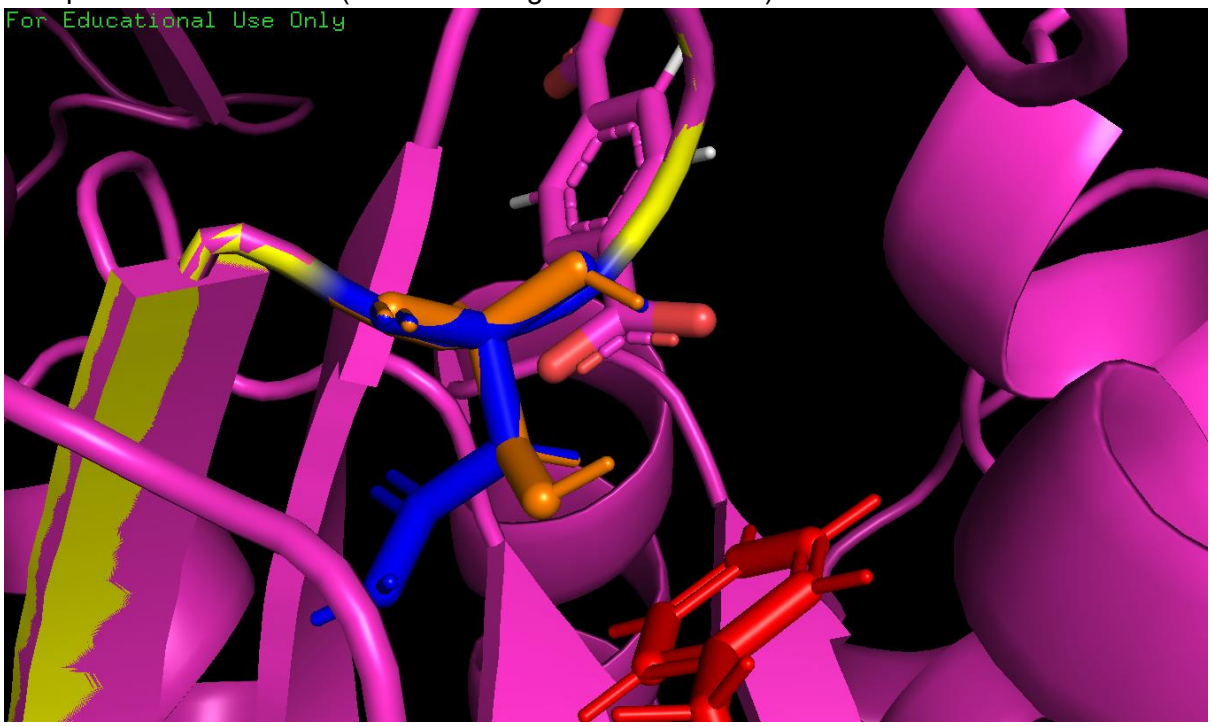


- Also compared how the changes actually look sterically and it seems that even though the mutations from Asn to Ser and Thr are smaller it does not change the size of the binding pocket since the Ser and Thr are smaller in the opposite directions.

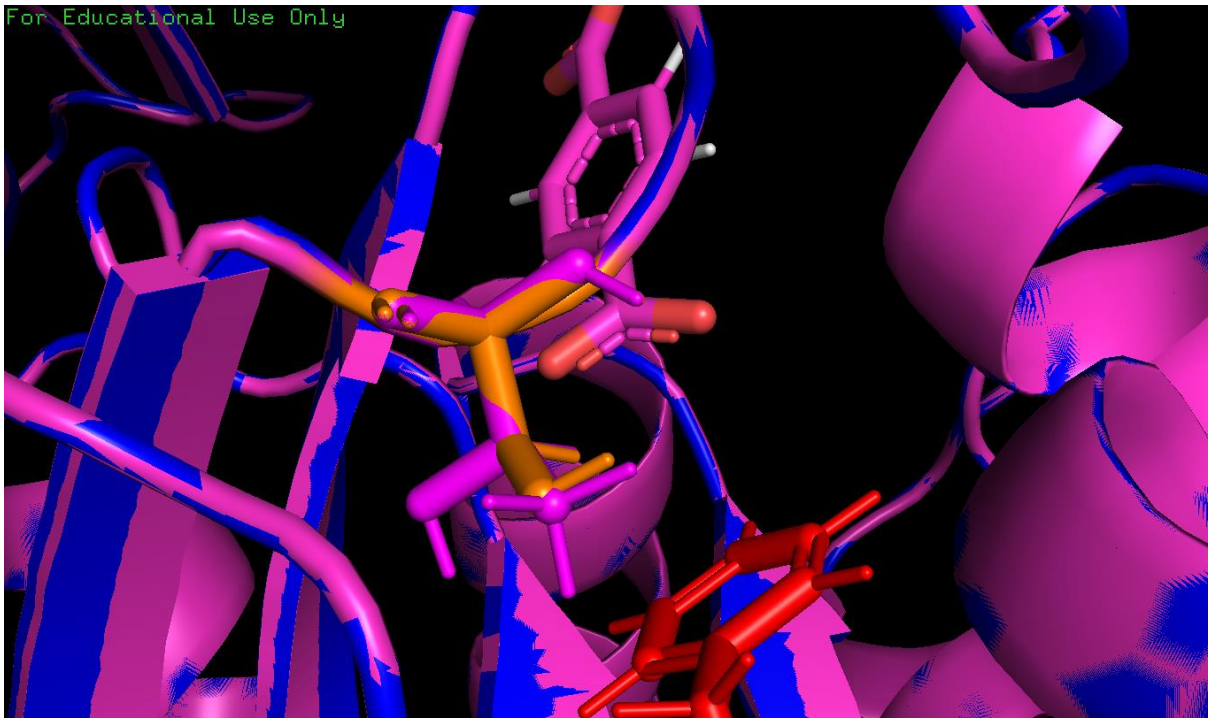
Only 311 = Serine (orange):



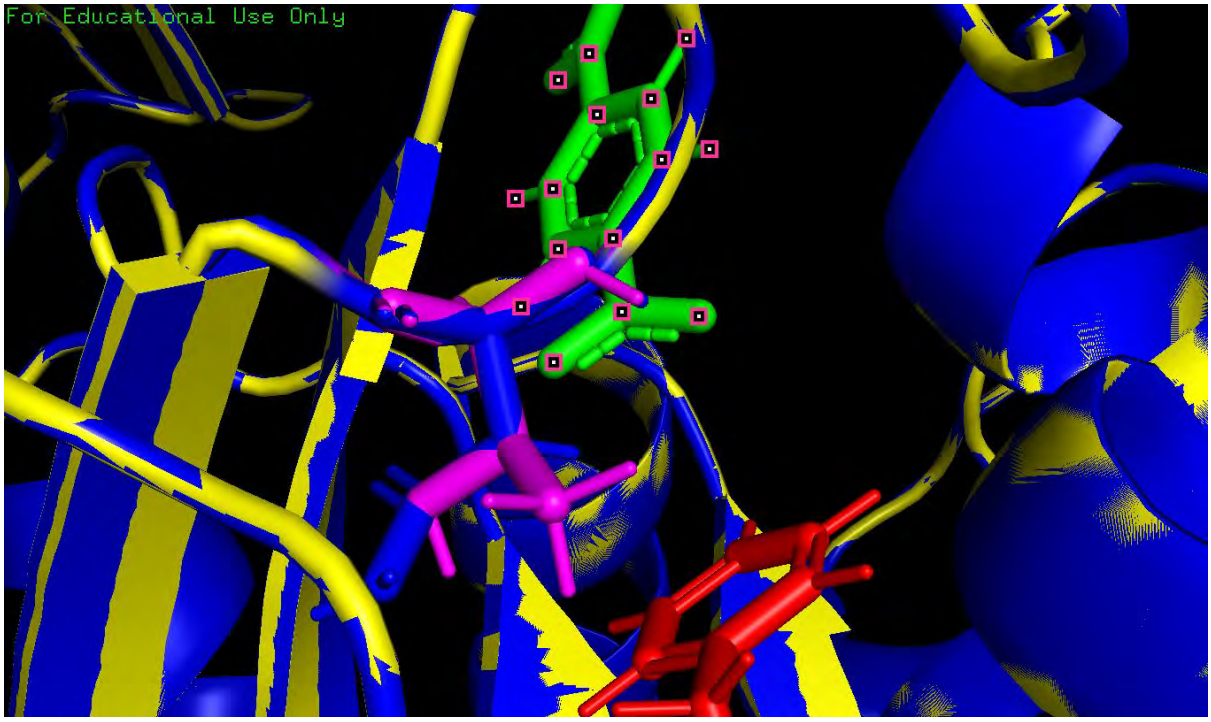
Comparison Ser and Asn (Serine = orange and Asn = blue):



Comparison Ser and Thr (Serine = orange and Thr = pink):



Comparison Asn and Thr (Asn = blue and Thr = pink):



- As can be seen in the images it seems the mutations makes the residue smaller but in the wrong direction, meaning that the side pointing to the ligand is relatively unchanged. Conclusion: these mutations do not change the binding site all that much.
- Is it possible to change Phe 184 → Tyrosine? Might give the substrate access to a positive hydrogen.
- Looked at Phe 184 → Tyrosine in pymol and it probably does not work.

Code

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

Mutation	Summary of Results
Asn 311 → Thr	Slightly worse binding affinity of similar position. Significantly worse ligand strain. Control docking with 4-chlorobenzoate gave similar results where the binding affinity was slightly worse. However, compared to the serine mutation the ligand strain was about the same, suggesting that the Threonine is a more conservative mutation in this case. Conclusion: Not a favorable mutation.
Asn 311 → Ser	Slightly worse binding affinity of similar position. Significantly worse ligand strain. Control docking with 4-chlorobenzoate gave similar results where the binding affinity was slightly worse and the ligand strain was significantly worse. Conclusion: Not a favorable mutation.

Interpretation

No favorable mutations found.

Next Steps

- Look up other potential mutations. Ask if you can mutate the Phe without destroying the tertiary structure. Show the steric pictures and discuss them.

2025-07-01 Exploring the effects of a point mutation on docking: 1T5D with Asn311

Contributors: Hilya, Sixten

Objective

Compare how 4-chlorobenzoate (4CB) and terephthalic acid binds to 4-chlorobenzoate ligase and the effect of mutating one of the residues in the active site to how the ligand sits inside the binding pocket.

Background

One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

We want to try a few mutations:

Asn311→Trp

Mutation to a very big hydrophobic residue. Would be interesting to see if this "destroys" the binding, confirming that Asn311 is important for substrate binding.

Asn 311 → His

Chlorine in 4CB is uncharged, but the other carboxylic acid group in TPA is negatively charged, so it might be interesting to see how a positively charged residue interacts with the different substrate.

Asn 311 → Lys

Same reason as mutation to His.

Methods and parameters

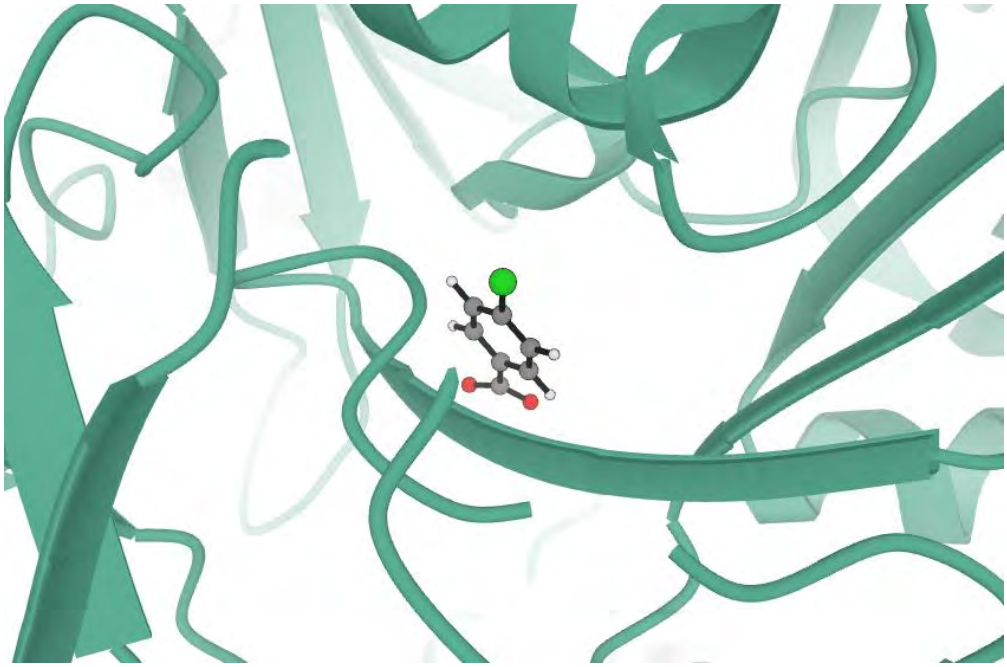
Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), ChimeraX
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures:

	4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

Control: 1T5D with 4-Chlorobenzoate

Starting first with the control, redocking 1T5D in Rowan with the original ligand.
Files: used 1T5D here RCSB PDB - 1T5D: 4-Chlorobenzoyl-CoA Ligase/Synthetase bound to 4-chlorobenzoate and SDF file from 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem

Pocket saved, sanitised protein, and then redocked with SDF file.



"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.209","0.9720123703714504"
"2","-2.977","4.7383249895505895"
"3","-2.648","0.013177701566729502"

N311→W

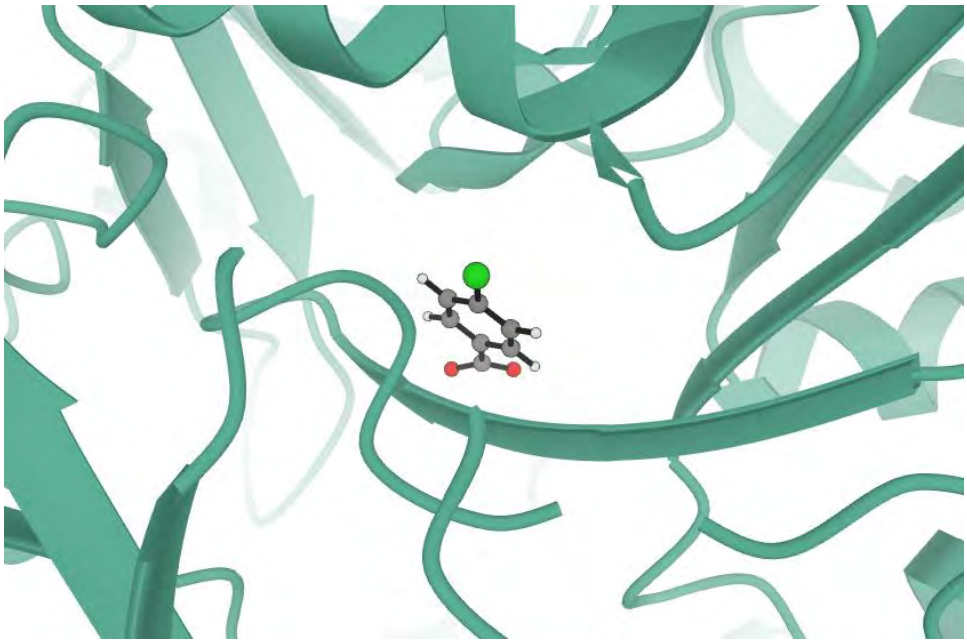
Original Substrate: Mutated 1T5D (N311→W) with 4-Chlorobenzoate

Mutated N311 to W in PyMOL.

PDB file here:
https://drive.google.com/file/d/1ILD7FDO420JaejWI3ytZZMEXhQsgZNo8/view?usp=sharing

Redocked in Rowan using the same SDF file.

Results
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.225","1.4733925407896997"
"2","-2.984","4.8920648415904715"
"3","-2.64","0.0320029895905398"
"4","-2.618","0.913026468049036"

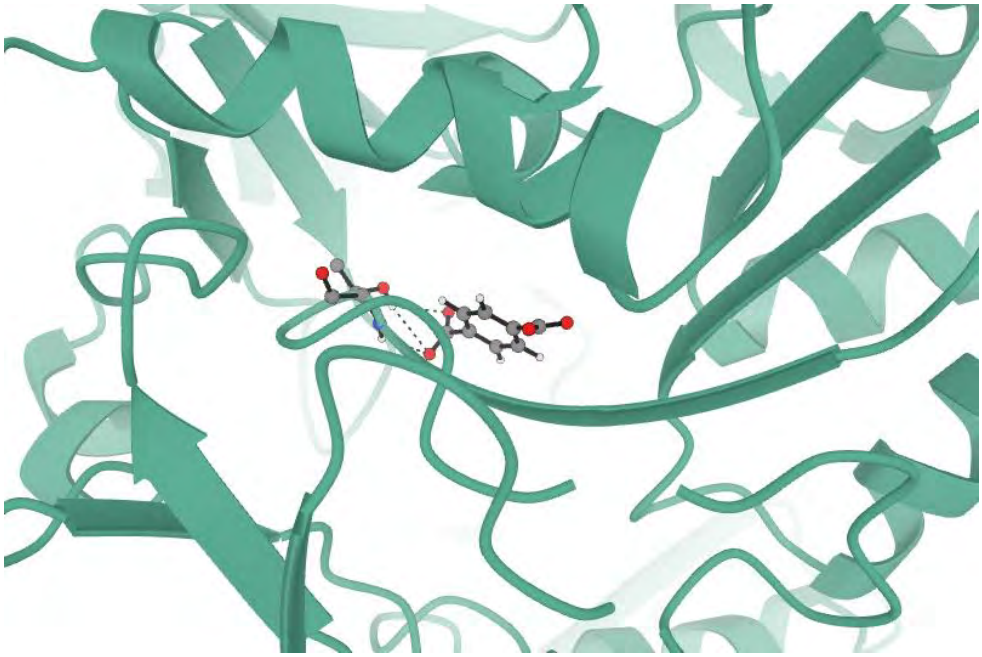


Tbh pose looks kind of the same….

New Substrate: Mutated 1T5D (N311→W) with TPA

Used the same PDB file (sanitised) to redock TPA with the mutated enzyme.
SDF file for TPA: Terephthalic acid | C8H6O4 | CID 7489 - PubChem, but removed the hydrogen atoms in Rowan and changed charge from 0 to -2.

Results
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-2.904","7.338097262350037"
"2","-2.683","2.8332058439808008"
"3","-2.679","2.7353143464139347"
"4","-2.64","2.828813276791891"



Actually some hydrogen bonds appeared?

N311→H

Original Substrate: Mutated 1T5D (N311→H) with 4-Chlorobenzoate

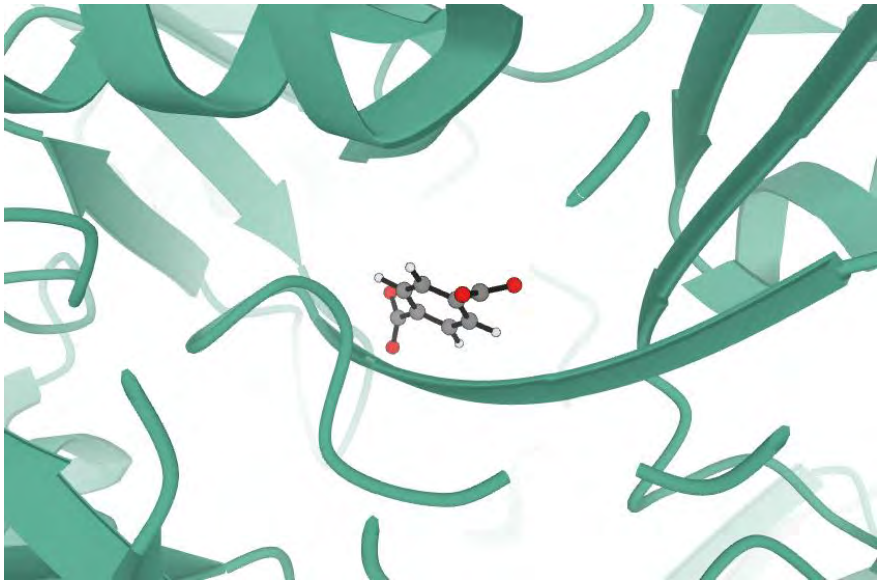
Results

"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.264","1.2412139888051774"
"2","-2.978","4.941010590409574"
"3","-2.627","0.013177701638069084"
"4","-2.621","1.2920422664195277"

New Substrate: Mutated 1T5D (N311→H) with TPA

Results

"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.264","1.2412139888051774"
"2","-2.978","4.941010590409574"
"3","-2.627","0.013177701638069084"
"4","-2.621","1.2920422664195277"



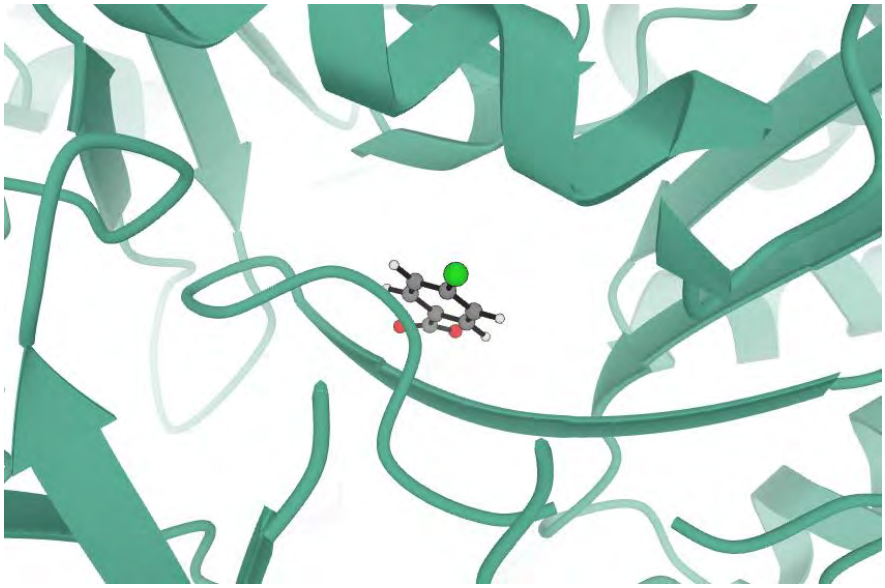
Same with the other mutation (N→W), seems to adopt a different pose.

N311→K

Original Substrate: Mutated 1T5D (N311→K) with 4-Chlorobenzoate

Results

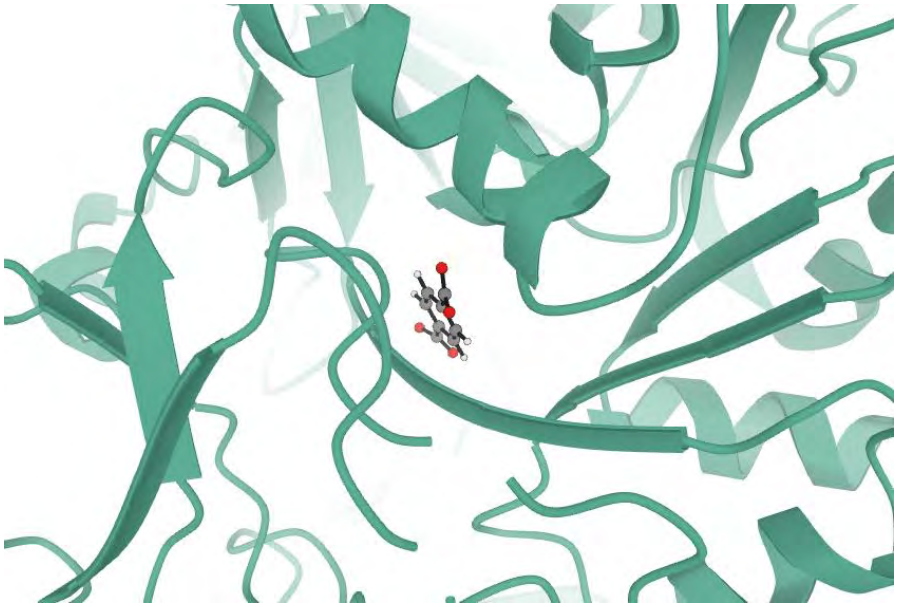
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.228","0.7668167311901426"
"2","-2.977","4.599645367944024"
"3","-2.569","2.2471118775624146"
"4","-2.275","4.767190431149051"



New Substrate: Mutated 1T5D (N311→K) with TPA

Results

"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-2.926","0.025100384007967203"
"2","-2.902","2.570906831200986"
"3","-2.888","2.11847240960184"
"4","-2.417","1.2098385088130534"



Seemed to be more correct!!! More TPA-like

File:
https://drive.google.com/file/d/1HbLU0fGRlBtJl5lN7LJsT6DhCgsqtz_U/view?usp=drive_link

Code

No scripts used. Everything was performed using Rowan GUI.

Results Summary

Mutation	Summary of Results
Asn 311 → Trp	Comparing the chlorobenzoate docking before and after the mutations…. didn't change that much! Worse ligand strain, but energy change is not significant. When docking TPA the score changed from -3.2 to -2.9 so it's not a "very ugly" mutation after all? The ligand strain is *really heavy (+7 kcal/mol)* and the position really changes. I think this only confirms that Trp crowds the pocket but not much else. Conclusion: Not a favorable mutation, but perhaps Asn311 is not as crucial as we previously thought?
Asn 311 → His	On the docking with the original substrate: increased the ligand strain, maybe because histidine is larger with the ring, compared to asparagine. But the score actually decreased (good) by ~ 0.05.

	However, similarly like the 311N→W mutation the ligand adopts a different pose, and has huge ligand strain. Binding energy is comparable with 311N→W (-2.9), but no H-bonding this time. Conclusion: Not a favorable mutation for TPA, and again I don't think Asn311 is something really important! Though we might mix and match it with other mutations.
Asn 311 → Lys	For 4-chlorobenzoate it actually *reduces* the strain and docking score! Really good ligand strain for TPA, even *better than 4-chlorobenzoate.* Pose seems to be more correct. Binding score got worse but *slightly* better than others mutations (-0.02). No hydrogen bond. Conclusion: A potentially good mutation

Interpretation

We think that the Asn311 → Lys might be a favorable mutation. Other details please see the summary table above.

Next Steps

Test more mutations with this workflow and come up with ideas on other different methods to test mutations.

2025-07-01 Exploring the effects of a point mutation on docking: 1T5D with Asn311, part 2

Contributors: Sixten, Hilya

Objective

Compare how 4-chlorobenzoate (4CB) and terephthalic acid binds to 4-chlorobenzoate ligase and the effect of mutating one of the residues in the active site to how the ligand sits inside the binding pocket.

Background

(See entry for 2025-07-01) One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

Mutations to test and rationale:

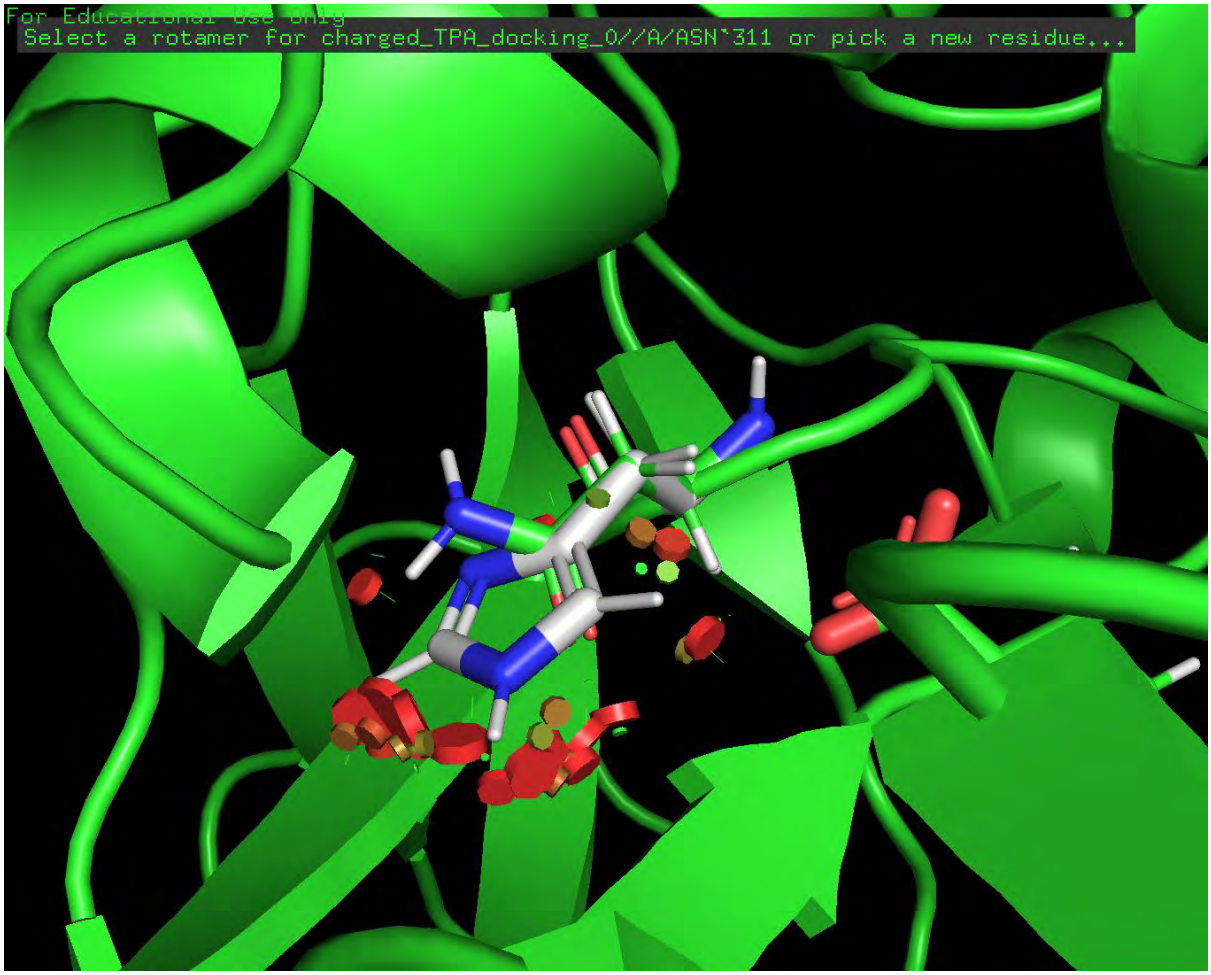
Asn 311 → Gly	Much smaller, and also in the correct places.
Asn 311 → His	Chlorine in 4CB is uncharged, but the other carboxylic acid group in TPA is negatively charged, so it might be interesting to see how a positively charged residue interacts with the different substrate.

Methods and parameters

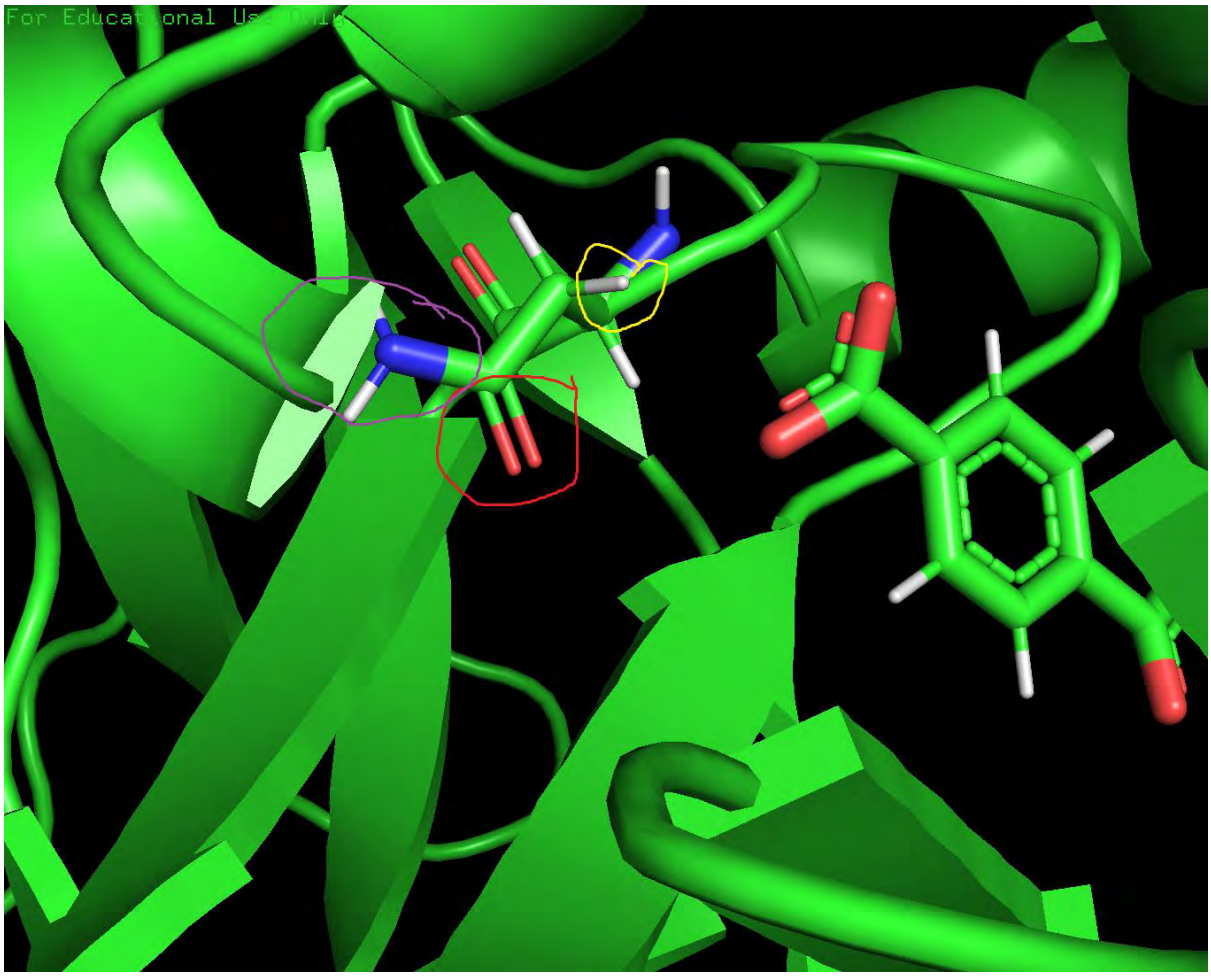
Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), PyMOL
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D

	Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

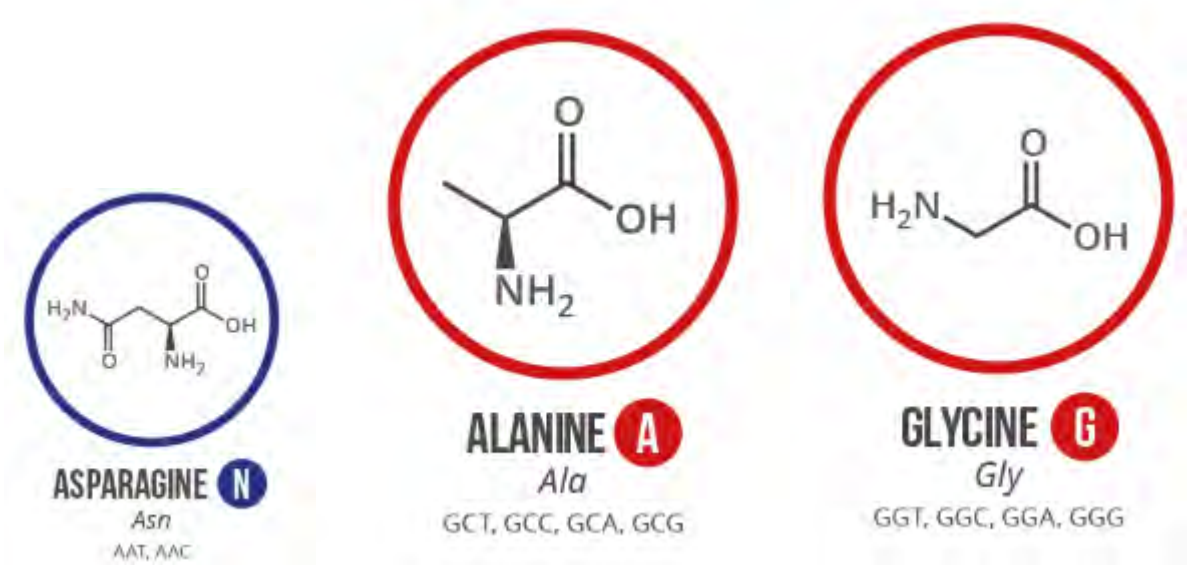
- Looked at how the Asn 311 → His looks. Does not look super good. The rotation of the residue in relation to the binding site is interesting. I do not think it is the outermost atom that interacts with the substrate but rather atoms in the middle of the side chain. As you can see there are alot of sterical problems with the His mutation. It also appears that the nitrogen atoms are pointed away from the substrate.



- From the image below it is possible to reason that it is the oxygen (red) and the hydrogen (yellow) that is the closest to the substrate.



- This is interesting because it means that the mutation should be focused on changing these parts of the side chain. Suggestions are: Asn → Gly or Ala



- As we can see they are much smaller. I am however unsure how the binding interactions change.
- Performed docking with mutated Asn 311 → Gly. Similar results to earlier, i.e slightly worse binding affinity and similar ligand strain. The unchanged ligand strain is interesting since it gives us more clues on how the steric interactions behave. Maybe it can be concluded that the Asn residue is not super important for binding, however, this is unclear.

Code

Docking was performed using Rowan GUI. Visual inspection was performed with PyMOL GUI.

Results Summary

Asn 311 → Gly	Worsens the binding, strain is a little worse. Control docking with 4-chlorobenzoate gave slightly worse binding affinity and similar ligand strain. Conclusion: Not a good mutation.
Asn 311 → His	On the docking with the original substrate: increased the ligand strain, maybe because histidine is larger with the ring, compared to asparagine. But the score actually decreased (good) by ~ 0.05. However, similarly like the 311N→W mutation the ligand adopts a different pose, and has huge ligand strain. Binding energy is comparable with 311N→W (-2.9), but no H-bonding this time. Conclusion: Not a favorable mutation for TPA, and again I don't think Asn311 is something really important! Though we might mix and match it with other mutations.

Interpretation

No favorable mutations.

Next Steps

Test more mutations with this workflow and come up with ideas on other different methods to test mutations.

2025-07-02 How does ligand binding change the structure of 1T5D? TPA vs. 4-chlorobenzoate

Contributors: Sixten

Objective

Investigate differences caused by 1T5H binding TPA or 4CB.

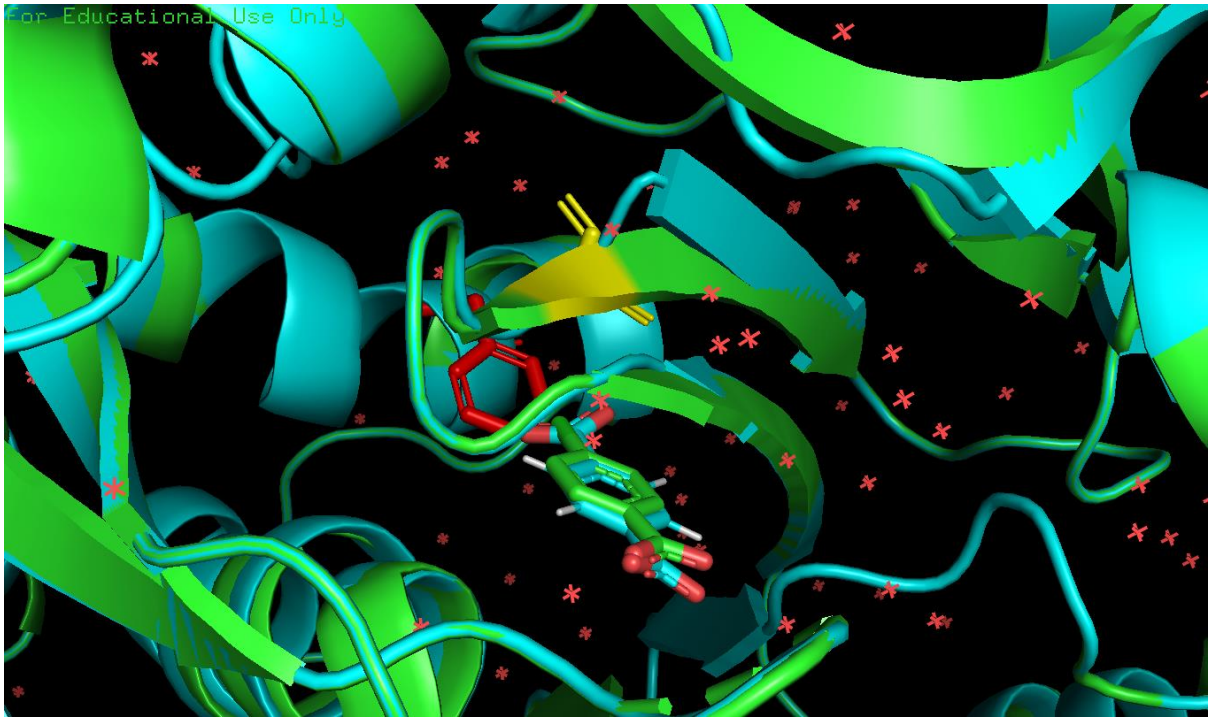
Background

(See entry for 2025-07-01) One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), ChimeraX
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

- Realized from yesterday's meeting that using the redock function in Rowan might be better. Should also use the original 1T5D PDB file for mutations (have used one where TPA was docked. Since it was sanitized this should be fine but you never know).
- When investigating the point above I found something interesting:



- Looked in the pdb file and it seems that there are two hydrogens and an oxygen missing form the file. This is weird. Maybe the oxygen is missing because it is part of the backbone?
- Yes, I am dumb. 14 is the correct number.

Code

None used.

Results Summary

- The original 1T5D PDB is very different from the one with docked TPA, especially at the place where Asn 311 is supposed to be. In the original file the Asn 311 residue is part of the beta strand (yellow) but is not in the docked file.

Interpretation/Discussion

- I propose an hypothesis for the difference. When looking in the pdb files for each of the structures the 311 residue in each of them is correctly Asn. However, in the docked file, the Asn contains more atoms (14) compared to the Asn residue of the original pdb file (8). The correct amount of 17 is in neither and I guess it just shows how little I understand of what is actually going on 😕. The idea is at least that the original pdb file is missing atoms since it is based on an experimentally determined crystalline structure. This is known to be a problem, especially in areas where the protein is flexible, i.e usually close to the binding site. The sanitation in rowan adds

these missing atoms, explaining the difference. Why Rowan does not add all of the atoms (17) is not entirely clear, but it is most likely some hydrogens missing.

Next Steps

- Look into the difference between docked TPA with original 1T5D.
- The question is now, which PDB file should be used to make the mutations? The binding site is definitely distorted in the original PDB file, so it does make it more difficult to determine what residues are relevant.
- Deciding to move forward using the sanitized version from Rowan. It appears it is better if all atoms are present when doing the mutations.

2025-06-08 Docking of TPA on mutants of 1T5D (Phe184) and notes

Contributors: Sixten, Hilya

Objective

Find the best potential mutations for 1T5D. Testing one of the potential mutations, which is Phe184.

Background

One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

We identified some potential residues to potentially mutate: Phe184, Asn311, Ile303, and combinations of those.

We want to focus on the Phe184.

Mutation	Rationale
Phe 184 → His	Phe ring faces towards the carboxylic acid group of 4-chlorobenzoate so it might be interesting to see how charge affects this. Want to keep the ring shape.
Phe 184 → Ser	Changing into a smaller residue. Trying to see how a smaller residue affects the ligand binding.

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), ChimeraX

Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

Methods look at entry 2025-07-01.

Summary of Results

- Putting focus on the Phe 184 residue. We will begin to mutate and dock histidine and serine.
- Not able to use the redock function for some reason, but using Hilya's coordinates should be sufficient.
- Did docking with both redock and manual. Gave quite different results. The redock did not manage to bind the TPA in the binding pocket while the manual docking did.
- Phe 184 to His gave the pocket position to -2,8 binding affinity and ligand strain of 1,128.

Mutation	Summary of Results
Phe 184 → His	Not better; higher strain and lower binding affinity (slightly). Similar results when docking 4-chlorobenzoate, but not a significant change in binding affinity. Conclusion: Not a good mutation.
Phe 184 → Ser	Not better; higher strain and lower binding affinity (slightly). Similar results when docking 4-chlorobenzoate, but not a significant change in binding affinity. Conclusion: Not a good mutation.

Interpretation

- After performing the dockings with the new mutations it appears that the Phe aromatic ring is important for binding.

Next Steps

- Trying to find exactly what residues are the closest to the carboxyl group of TPA.
- Thinking we should have a (positive or negative do not know name) control that dock a substrate known to have no affinity for the enzyme.

2025-07-05 Docking of TPA on mutants of 1T5D (Ile303)

Contributors: Sixten

Objective

- Investigate exactly what residues and atoms are close to the carboxylic group of the TPA.
- Investigate effects of Ile303 mutation to ligand strain and affinity

Background

After learning about residues in the active site we come up with this mutation to test:

Mutation	Rationale
Ile 303 → Ser	Making the residue smaller and also giving the opportunity for the bonding with a hydrogen with a partial charge.

Methods and parameters

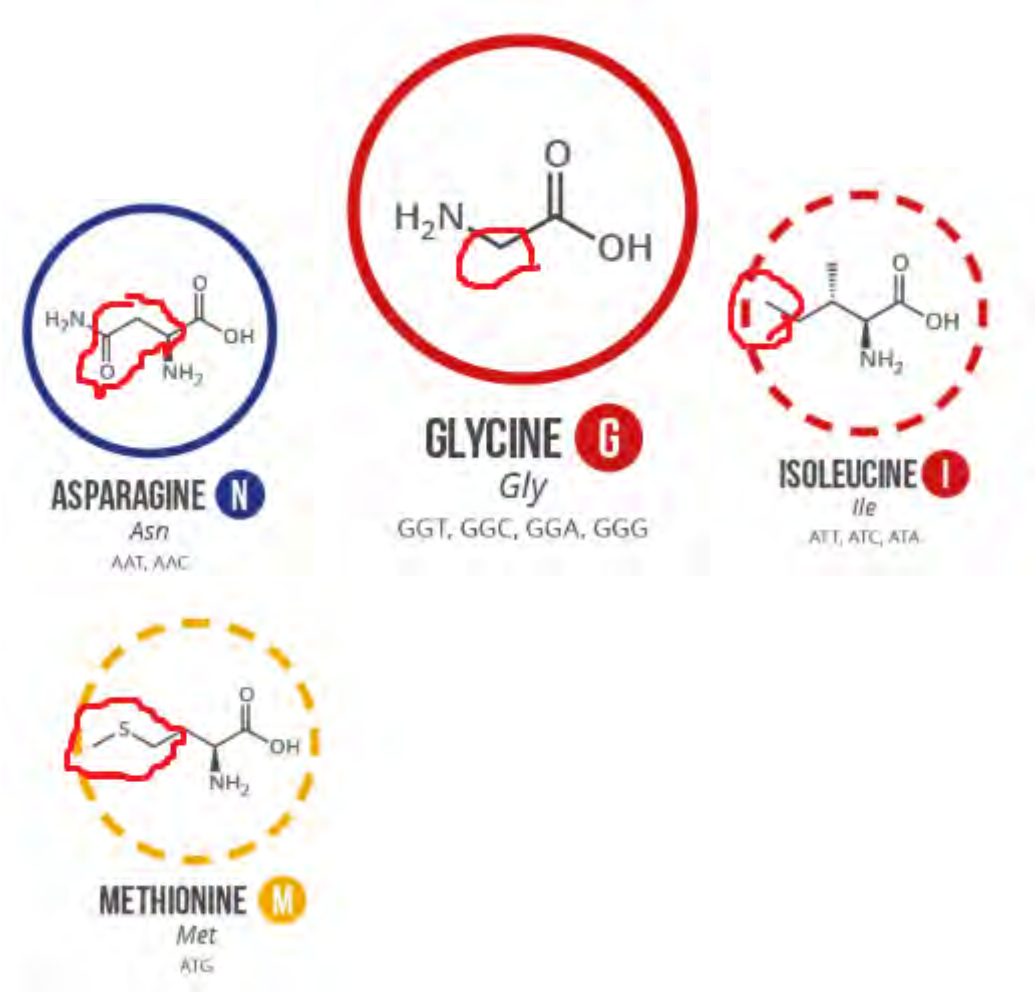
Component	*Description*
Software Used	Rowan, PyMOL
Input Structure	TPA, 1T5D
Key Parameters	Binding affinity, ligand strain, ligand position and orientation, ligand steric clashes with protein
Script Location	Compare atoms within a certain distance

- Residues within 3Å of carboxylic acid group of TPA:

Residue (nr and name)	Atom (name)
Ile 303	CD1 + HD11 + HD12
Gly 305	HA3

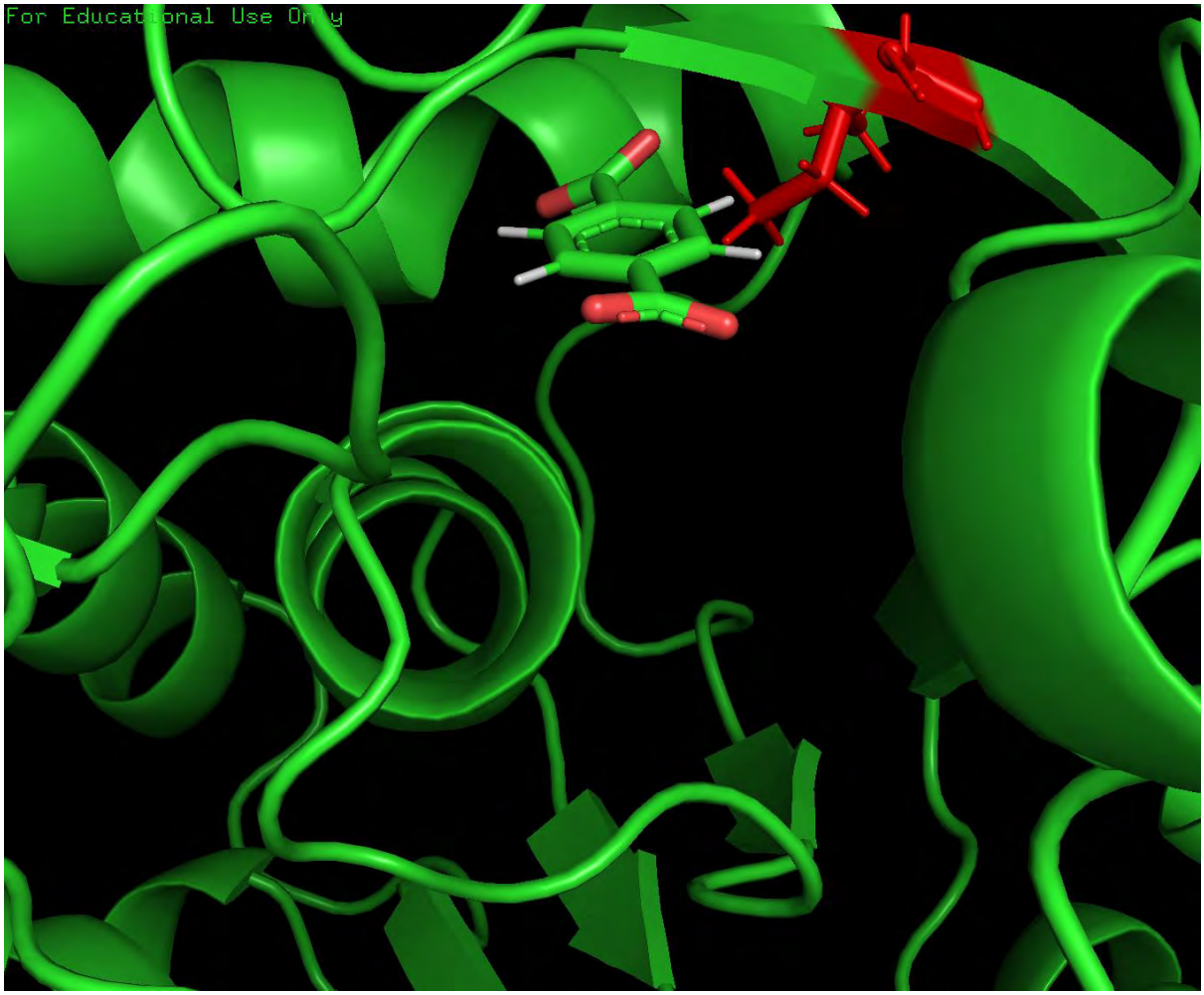
Met 310	CE + HG2 + HE3
Asn 311	HA + HB2

- Trying to highlight the part of the residue that is the closest to the carboxylic group:

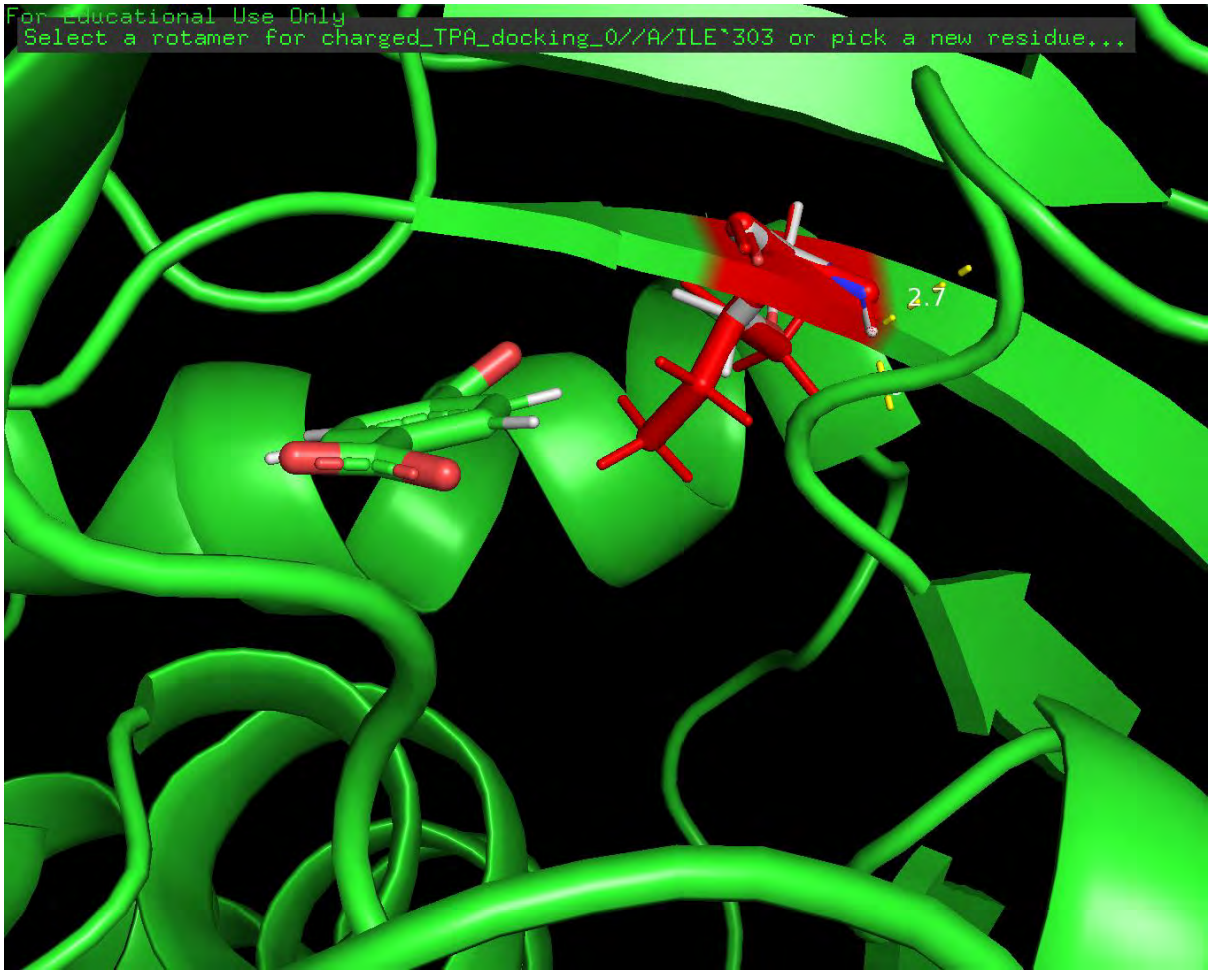


- Tried to make a movie in pyMOL, did not succeed very well due to issues when downloading it.
- Tried to focus on Isoleucine since it is the biggest, little polarity and methionine we should probably not touch due to the sulphur perhaps being important for catalytic activity or smth.
- Mutated isoleucine to Ser due to Serine being smaller and having the ability to supply hydrogen bonds. Also made two different rotamers of the residue to see how this affects the docking. They are named r1 and r3.

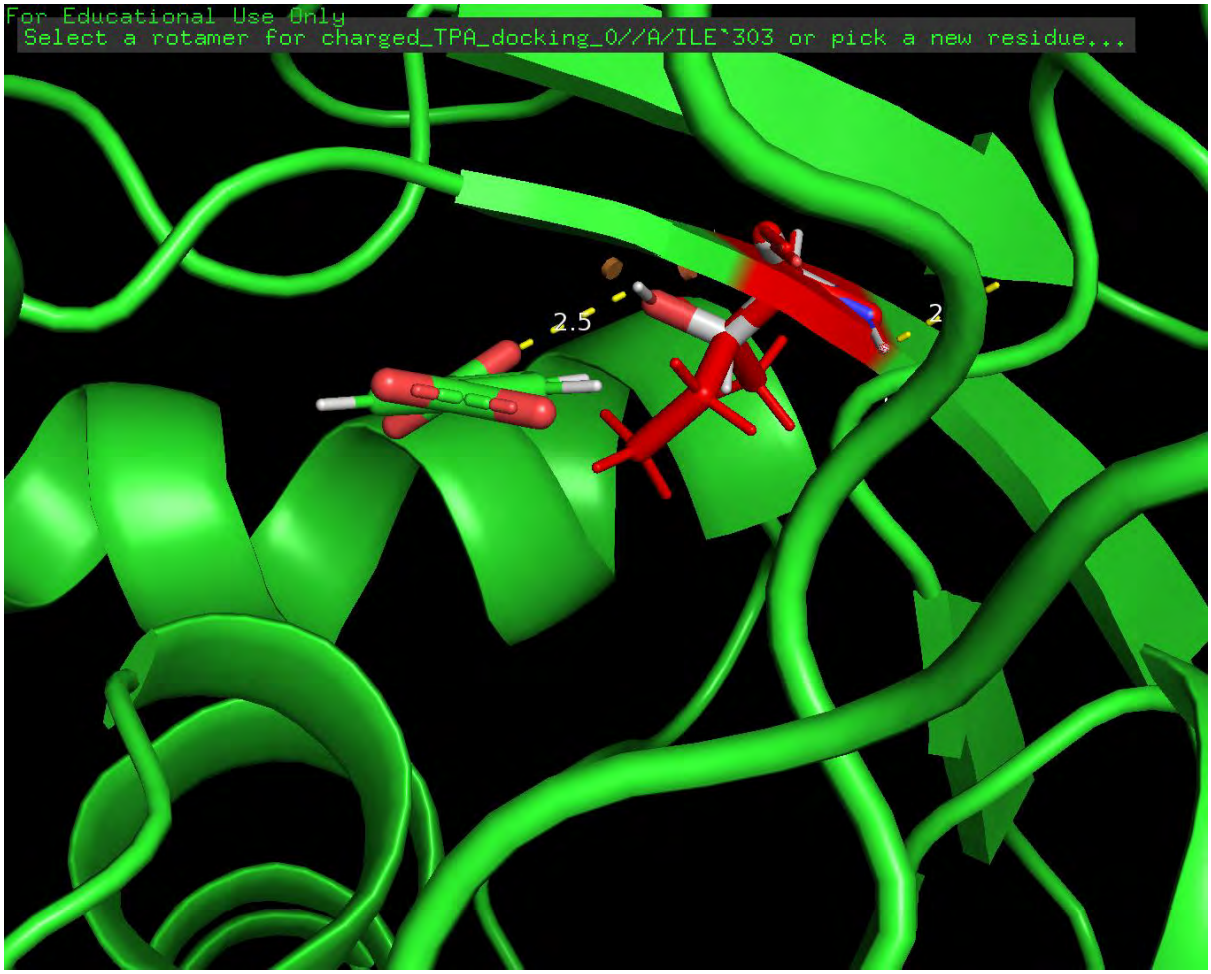
How Ile 303 looks:



Mutated rotamer 1:



Mutated rotamer 3:

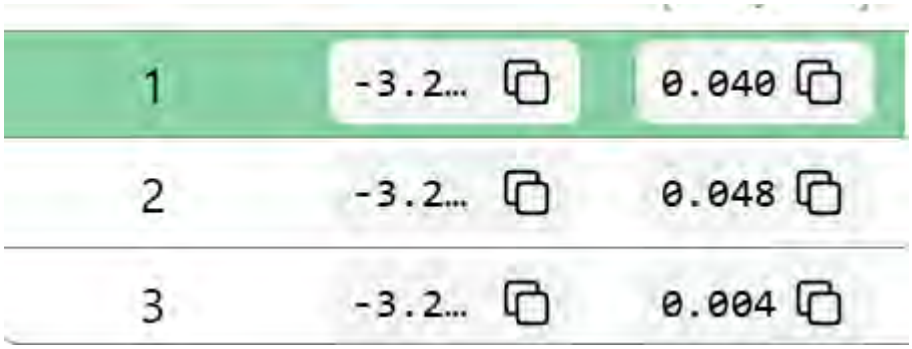


- Accidentally ran a docking with a TPA without the carboxylic hydrogens and a net charge of zero. Apparently this does not work and gives error :)
- Both r1 and r3 gave really good results!!!!

r1:

1	-3.3...	2.740
2	-3.3...	0.015
3	-3.2...	0.173

r3:



1	-3.2...	0.040
2	-3.2...	0.048
3	-3.2...	0.004

Code

```
#After smaking two selections, one with residues close to ligand A and one
#with residues close to ligand B, this script compares the selections
#and highlights differences

from pymol import cmd

def get_atoms(sel):
    atoms = set()
    for a in cmd.get_model(sel).atom:
        atom_id = f"{a.chain}/{a.resi}/{a.resn}/{a.name}"
        atoms.add(atom_id)
    return atoms

atoms1 = get_atoms("near_TPA3")
atoms2 = get_atoms("near_chloro3")

only_in_1 = atoms1 - atoms2
only_in_2 = atoms2 - atoms1

print("Atoms only in near_TPA3:")
for atom in sorted(only_in_1):
    print(atom)

print("\nAtoms only in near_chloro3:")
for atom in sorted(only_in_2):
    print(atom)
```

Results Summary

Mutation	Summary of Results
Ile 303 → Ser	Really good results when docking TPA. Binds very similar to original binding and gives better binding affinity and very low strain. Did for two different rotamers of Ser and gave very similar result. Control docking with 4-chlorobenzoate gives worse binding affinity and higher strain. That is good. Conclusion: Very good mutation it seems like.

Interpretation

Ile 303 → Ser could be a potential mutation to test further.

Next Steps

- I would want to perform the Gly mutation as well to see if it is the sterical change that is good or if it is the access to another hydrogen bond. Also need to perform the control docking with 4-chlorobenzoate

2025-07-09 Exploring the effects of a point mutation on docking: 1T5D with Asn311, part 2 + Ile303

Contributors: Sixten, Hilya

Objective

Compare how 4-chlorobenzoate (4CB) and terephthalic acid binds to 4-chlorobenzoate ligase and the effect of mutating one of the residues in the active site to how the ligand sits inside the binding pocket.

Background

One of the aspects that can contribute to enzyme activity is how well the substrate can be taken up by/bind the enzyme. If the substrate has a low affinity for the enzyme or has trouble accessing the binding pocket, it will negatively impact the reaction. For this case, we want to use the enzyme on a non-native substrate (TPA) with a similar functional group (carboxylic acid group) to the native substrate (4CB). If the new substrate doesn't sit in the correct orientation within the enzyme, this could be an indication that the enzyme will work poorly on the new substrate.

We want to try a few mutations:

Asn311→Thr

Thr and Ser has similar properties to Asn (polar uncharged), but a smaller. We hope that this can reduce the steric pressure on TPA (which is bigger due on that side to the carboxylic acid).

Asn 311 → Ser

Thr and Ser has similar properties to Asn (polar uncharged), but a smaller. We hope that this can reduce the steric pressure on TPA (which is bigger due on that side to the carboxylic acid).

Asn 311 → Gly

We want to look how mutating into a much smaller residue will affect the binding, and also Asn and Gly both have an amine group.

Ile303 → Gly

Making only smaller residue without the partially charged hydrogen to see if that is what is important or it is only the sterical changes that are favorable.

Methods and parameters

Component	*Description*

Software Used	Rowan (https://labs.rowansci.com/), ChimeraX
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

Methods look at entry 2025-07-01.

Code

No scripts used. Everything was performed using Rowan GUI.

Results summary

Mutation	Summary of Results
Asn 311 → Thr	Slightly worse binding affinity of similar position. Significantly worse ligand strain. Control docking with 4-chlorobenzoate gave similar results where the binding affinity was slightly worse. However, compared to the serine mutation the ligand strain was about the same, suggesting that the Threonine is a more conservative mutation in this case. Conclusion: Not a favorable mutation.
Asn 311 → Ser	Slightly worse binding affinity of similar position. Significantly worse ligand strain. Control docking with 4-chlorobenzoate gave similar results where the binding affinity was slightly worse and the ligand strain was significantly worse. Conclusion: Not a favorable mutation.

Mutation	Summary of Results
Asn 311 → Gly	Worsens the binding, strain is a little worse. Control docking with 4-chlorobenzoate gave slightly worse binding affinity and similar ligand strain. Conclusion: Not a good mutation.
Ile 303 → Gly	Really interesting results. When docked in similar position as original substrate the binding affinity are about the same as Ile 303 → Ser mutation (-3,2 and low strain). However, it seems the Gly mutation gives the TPA access to a position parallel to the beta sheet close to it, giving very high binding affinity (-3,6) and low strain. This probably means that it is a bad mutation since the TPA most likely needs to keep its orientation if the mechanism is supposed to work. Control docking with 4-chlorobenzoate gave worse docking in the original position compared to no mutation and also, as above, it got access to a position parallel to the beta sheet where the docking is very favorable. Conclusion: Probably a bad mutation.

Interpretation

No favorable mutations found. Other details please see the summary table above.

Next Steps

Test more mutations with this workflow and come up with ideas on other different methods to test mutations

2025-06-15 Notes on investigation of Ile303 mutations on 1T5D

Contributors: Sixten

Objective

The plan right now is to investigate the Isoleucine mutation a bit more.

Overview/Background

We have results for docking The Ile 303 → Ser with 4-chlorobenzoate. The mutation makes the docking results worse, which is positive. We also performed docking with Ile 303 → Gly mutation.

Results/Discussion

- Thought about making a protocol/explanation of how we do the docking. Maybe good for wiki/dry lab notebook.
- Mutation plus docking (In Pymol and Rowan):
    - Load the pdb file of the protein you would like to do a mutation on in Pymol.
    - Pymol has a wizard function that can do mutations for you. Simply select the residue you want to change and then pick a mutation. You can highlight the residue using Pymols select function to better see what the mutation does.
    - Download the mutated pdb file.
    - Upload the mutated pdb file to Rowan.
    - Use the sanitization function in Rowan to prepare the protein.
    - Upload the ligand and make appropriate changes to it (e.g for TPA you should remove the carboxylic hydrogens and change the charge to -2).
    - Select a box, either by using the redock function or choose the box manually
    - Run the docking and evaluate the results.
- Could start investigating the methionine mutations, look at coumarate or start looking at solubility in ethylene glycol.

Next Steps

- I think for now we have a lot of material on the 1T5D mutations and while we could explore more maybe it is more efficient to focus on some other area.
- Will begin with solubility modeling. To do this, the plan is simply to use the function in Rowan that can measure this and then compare with literature.

2025-07-10 Investigating the binding site of 4-coumarate ligase

Contributors: Sixten

Objective

Determine active site of 3TSY. Visualise residues of interest in pyMOL.

Background

The proposed residues for binding for 4-coumate ligase: (according to Uniprot):

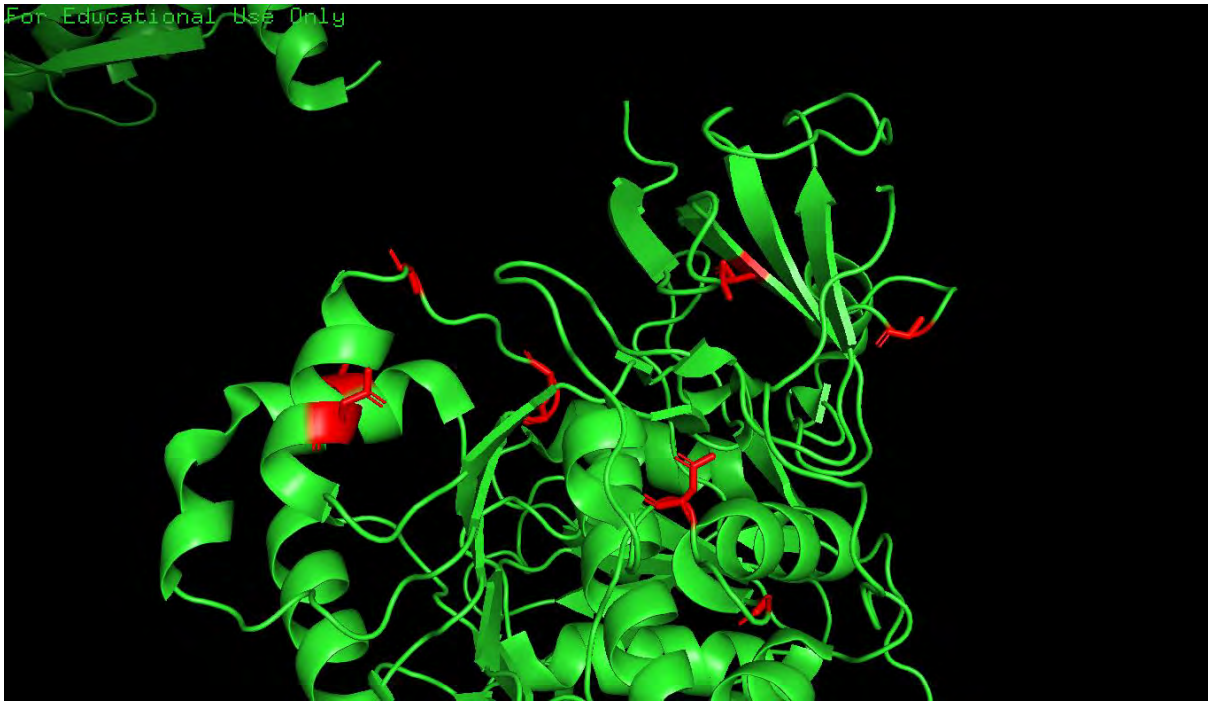
- 260
- 264
- 330
- 352
- 353
- 357
- 365
- 458
- 462

Methods and parameters

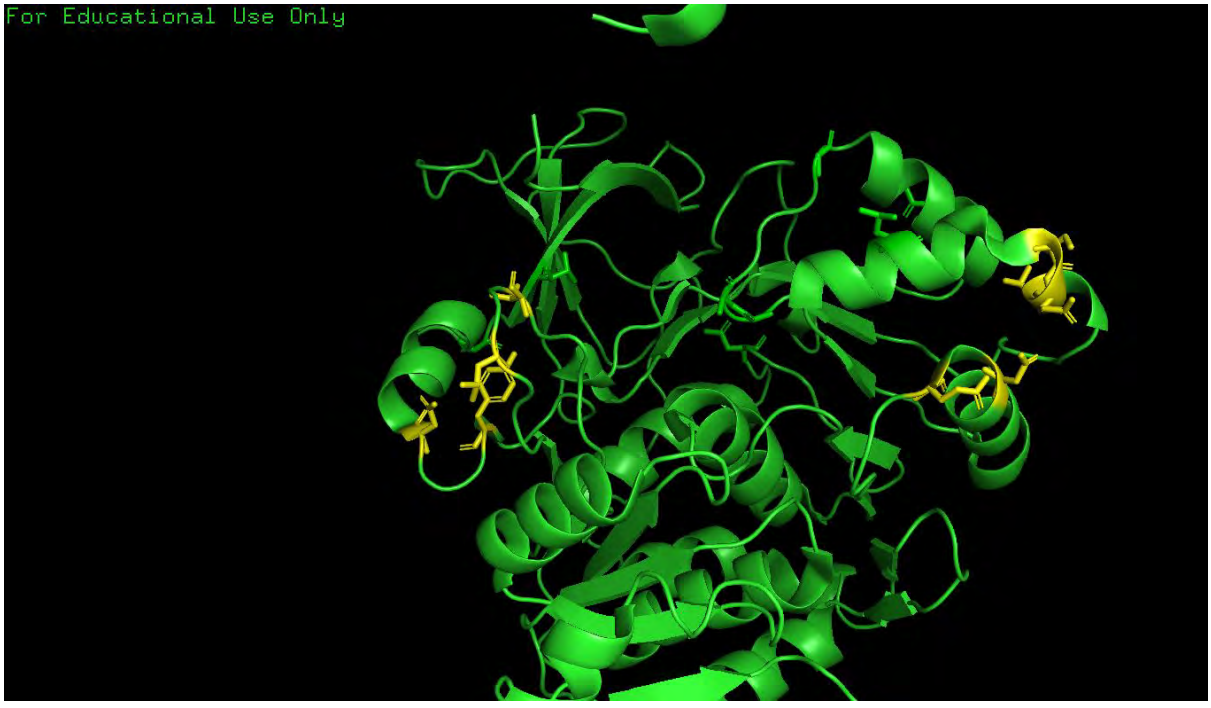
Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), pyMOL
Input Structure	PDB entry for 3TSY 4CL1 - 4-coumarate--CoA ligase 1 - Arabidopsis thaliana (Mouse-ear cress) | UniProtKB | UniProt Ligand; 4 coumarate
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

- After exploring the function in Rowan it seems like it needs the molecule to be uploaded at pubchem. I cannot find either PET or the enzymes on that website so I think we will keep that project for later.
- These seem to not actually work. Very much all over the place when highlighted in pyMOL and there is no defined binding pocket for the coumarate.

Important residues (according to Uniprot https://www.uniprot.org/uniprotkb/Q42524/entry) in red:



- According to Marios there are other residues of interest:
    - “Tyr-236/239, Gly-306/308, Pro-337, Val-338, Thr-336, Lys-523/526, Gln-443/446 and the catalytic Lys-540” (from Marios tab)
- Tried to put these into pymol and looks better but still some question marks. The active site seem to be split but now we can at least determine to possible pockets where the coumarate could fit.



Code

None used.

Results Summary

Still unclear where the ligand binds, but we can at least determine to possible pockets where the coumarate could fit.

Interpretation

None.

Next Steps

Try to find more structures of 4-coumarate ligase and more literature. Maybe try find a docked structure.

2025-07-13 Docking of coumarate to coumaroyl-CoA ligase with Rowan

Contributors: Sixten, Marios

Objective

Performing first docking with coumarate and 3tsy enzyme to get an idea of where the binding pocket is and see if it matches the theorized binding pocket Marios suggested.

Background

- UniProt entry Q9SMT7 – *Arabidopsis thaliana 4CL2*
    - Provides curated functional domain annotation for key residues involved in coumarate binding
- Ehlting J. et al. (2001)
  Identification of 4-coumarate:coenzyme A ligase substrate recognition domains
  Plant J, 27(5):455–465.
    - Uses domain swapping between At4CL1/At4CL2 to reveal regions that affect specificity, notably toward 4-coumarate vs. ferulate.
    - PubMed PMID: 11576429; DOI: 10.1046/j.1365-313x.2001.01122.x / https://pubmed.ncbi.nlm.nih.gov/11576429/
- Hu Y.-L. et al. (2010)
  Crystal structures of Populus tomentosa 4CL1 elucidate enzymatic mechanisms
  Plant Cell, 22(9):3093–3104.
    - Presents actual ligand-bound structures showing coumarate pocket with residues Tyr-236, Gly-306, Lys-523, etc.
    - PubMed link available; DOI: 10.1105/tpc.110.074054 / https://pubmed.ncbi.nlm.nih.gov/12819348/
- Liu G. et al. (2017)
  4CL involved in coumarin biosynthesis in Peucedanum praeruptorum
  Frontiers in Plant Science, 8:4.
    - Highlights biochemical roles of residues Tyr-239, Lys-441, Gln-446, Lys-526 in coumarate processing.
    - DOI: 10.3389/fpls.2017.00004 / https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2017.00004/full
- Schneider K. et al. (2003)
  The substrate specificity-determining amino acid code of 4-coumarate:CoA ligase
  Proc Natl Acad Sci USA, 100(14):8601–8606.
    - Defines the 12-residue “specificity code” for coumarate binding, showing adaptability to ferulate and sinapate with mutations.
    - PubMed PMID: 12819348; DOI: 10.1073/pnas.1430550100

Main papers to consider:

1. Schneider et al. (2003) paper (DOI:10.1073/pnas.1430550100), which defines the 12-residue "specificity code" for 4-coumarate binding in Arabidopsis 4CL2.
2. Hu et al. (2010) (DOI:10.1105/tpc.110.074054) provided the crystal structure of Populus tomentosa 4CL1 (PDB:3A9U), showing residues like Tyr236 and Lys523 interacting with the substrate.
3. Liu et al. (2017) (DOI:10.3389/fpls.2017.00004) confirmed critical residues (eg Lys441, Gln446) through mutagenesis.

Schneider et al. (2003, PNAS 100:8601–8606, DOI: 10.1073/pnas.1430550100), defines a 12 residue “signature code” showing how plant 4CL enzymes recognize substrates like p-coumarate. They show that residues such as Tyr-239, Gly-308, Thr-336 and the catalytic Lys-540 sit within about 6 Å of the bound coumarate.

Next, Hu et al. (2010, Plant Cell 22:3093–3104, DOI: 10.1105/tpc.110.074054) provided the crystal structure of *Populus tomentosa* 4CL1 (PDB 3A9U) with an AMP–substrate analog. From that I confirmed that residues Tyr-236, Gly-306, Pro-337, Val-338, Lys-523 and Gln-443 form the same pocket and align well with Schneider’s signature code.

Liu et al. (2017, Front. Plant Sci. 8:4, DOI: 10.3389/fpls.2017.00004) highlights Tyr-239, Met-306, Ala-309, Gly-334 and Lys-526 as critical for coumarate coordination, reinforcing the core pocket definition.

Putting these together, I settled on this s list of residues for *p*-coumarate binding: Tyr-236/239, Gly-306/308, Pro-337, Val-338, Thr-336, Lys-523/526, Gln-443/446 and the catalytic Lys-540 also **Phe-244 might be of interest.**

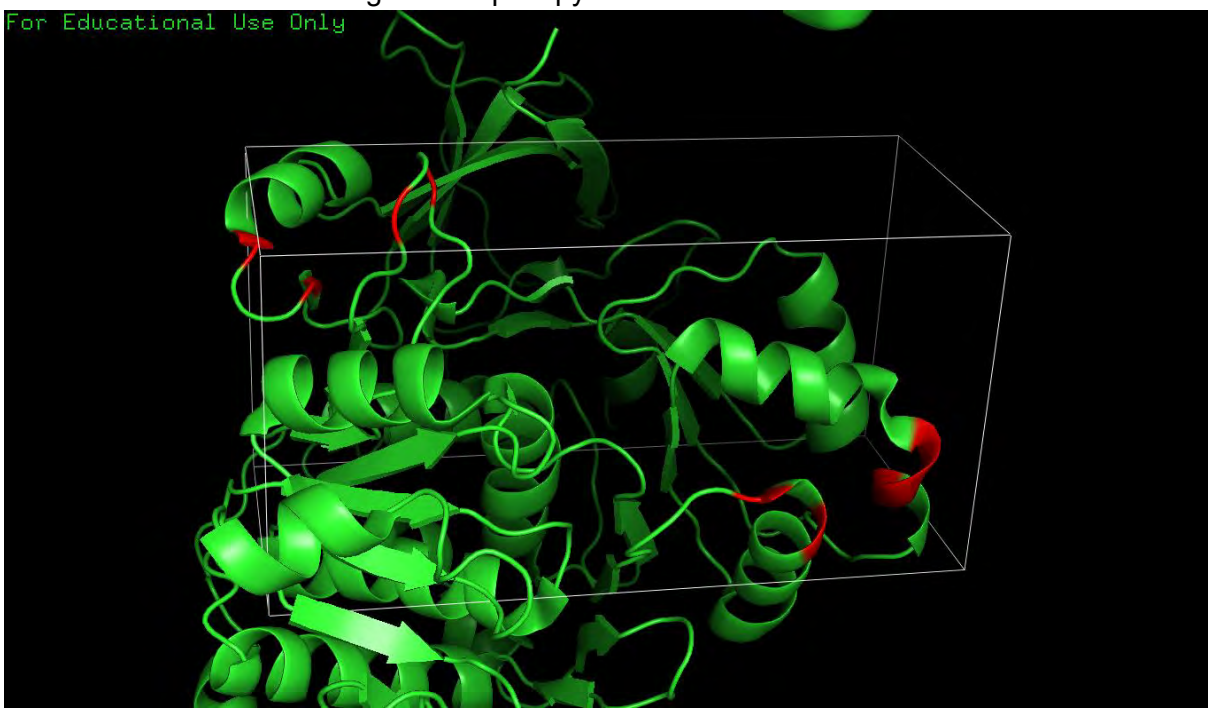
Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), pyMOL
Input Structure	PDB entry for 3TSY RCSB PDB - 3TSY: 4-Coumaroyl-CoA Ligase::Stilbene Synthase fusion protein Ligand 3D structure P-Coumaric Acid | C9H8O3 | CID 637542 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	Binding sites 3TSY

- NOTE: After sanitization in Rowan the protein looks really weird.

- NOTE: The 4-coumarate is charged at physiological pH. Remove hydrogen and add a charge in Rowan.
- The blind docking gave poor results, at least assuming the proposed active sites are somewhat correct.
- Proposed binding site is red. These are the two best dockings.

- Extracted a box using box script in pymol. Results:




```
Use these for docking:
  Center     = [78.35599899291992, -95.93199920654297, -37.9114990234375]
  Box size   = [49.09400177001953, 21.897994995117188, 25.388999938964844]
```

- Putting these into Rowan and running the docking again. Name of docking is box_docking.
- Results seems better with box. (NVM they dont). Still not really any connections to the proposed binding residues.

- Played around a bit in pyMOLI with the proposed active residues from Uniprot. Will make an appropriate box and see what docking results are given.
- Also, a comment on the changed structure when sanitizing in rowan is that we really do not know what it does.
- Docking with the Uniprot residues did not give better results (i.e the docked ligands or not in a defined pocket).

Code

```
#A function that lets you have several different selection available

from pymol import cmd
from pymol.cgo import BEGIN, LINES, COLOR, VERTEX, END

def sel_b(option):
    if option == '1':
        cmd.select('proposed_binding_site_1', 'resi
257+281+282+283+287+350+352+353+376+377+378+379+384+385+388')
        cmd.show('sticks', 'proposed_binding_site_1')
        cmd.color('yellow', 'proposed_binding_site_1')
    elif option == '2':
        cmd.select('proposed_binding_site_2', 'resi
213+237+238+239+243+306+308+309+332+333+334+335+336+340+341+344+441+444')
        cmd.show('sticks', 'proposed_binding_site_2')
        cmd.color('red', 'proposed_binding_site_2')
    elif option == '3':
        cmd.select('proposed_binding_site_3', 'resi 213+243+306+332+341+344')
        cmd.show('sticks', 'proposed_binding_site_3')
        cmd.color('red', 'proposed_binding_site_3')



cmd.extend('sel_b', sel_b)
```

To insert Code Block: Go to Add-ons → Code Blocks
This is a guide: https://www.geeksforgeeks.org/how-to-add-code-block-in-google-docs/

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

Ligand docked in the wrong position, even with a defined box.

Interpretation

3TSY might not be the correct structure to dock coumarate, e.g., is it in the wrong conformation, etc?

Next Steps

Figure out what sanitizing does in Rowan.

2025-07-14 Exploring the effects of a point mutation on docking: 1T5D with Phe184

Contributors: Hilya

Objective

Compare how 4-chlorobenzoate (4CB) and terephthalic acid binds to 4-chlorobenzoate ligase and the effect of mutating one of the residues in the active site to how the ligand sits inside the binding pocket.

Background

See background for entry 2025-07-09.

We want to do some mutations on Phe184.

Mutation	Rationale
Phe 184 → Thr	Changing into a smaller residue. Trying to see how a smaller residue affects the ligand binding.
Phe 184 → Lys	See how charge affects binding.

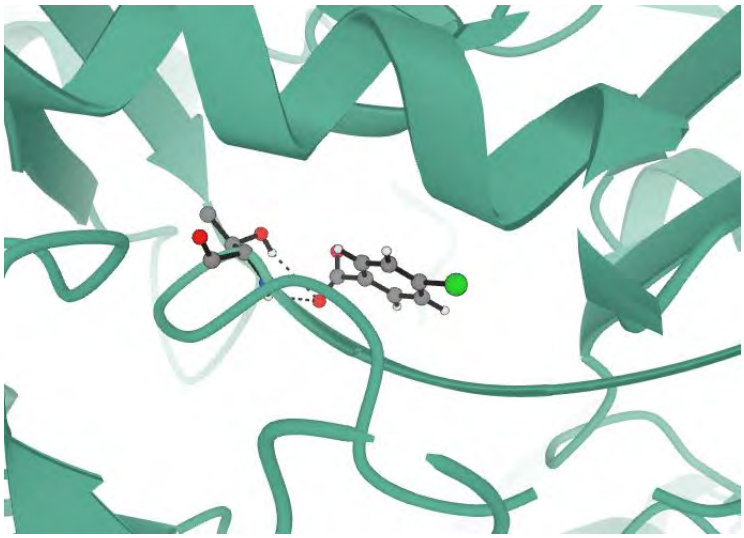
Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/)
Input Structure	PDB entry for 1T5D https://www.rcsb.org/structure/1T5D Ligand 3D structures: 4-Chlorobenzoate | C7H4ClO2- | CID 4359436 - PubChem
Key Parameters	Location of ligand within the protein, ligand strain, docking score
Script Location	No scripts used.

Original Substrate: Mutated 1T5D (F184→T) with 4-Chlorobenzoate

Results

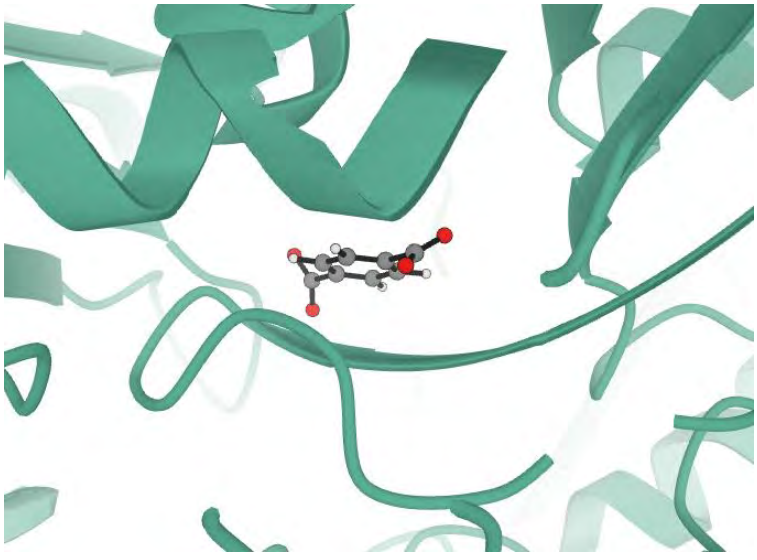
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-2.987","4.498616322385079"
"2","-2.973","0.10667663194468614"
"3","-2.628","0.8182725183315688"
"4","-2.58","2.6041648399883575"



New Substrate: Mutated 1T5D (F184→T) with TPA

Results

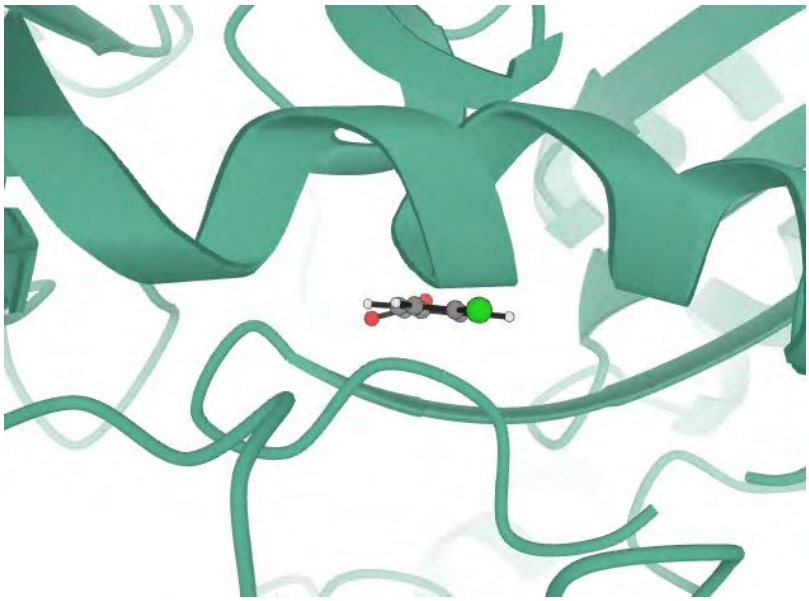
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-2.915","5.587345478418546"
"2","-2.682","2.7516295959727426"
"3","-2.681","2.8382259207681266"
"4","-2.67","3.0842096839891333"



Original Substrate: Mutated 1T5D (F184→K) with 4-Chlorobenzoate

Results

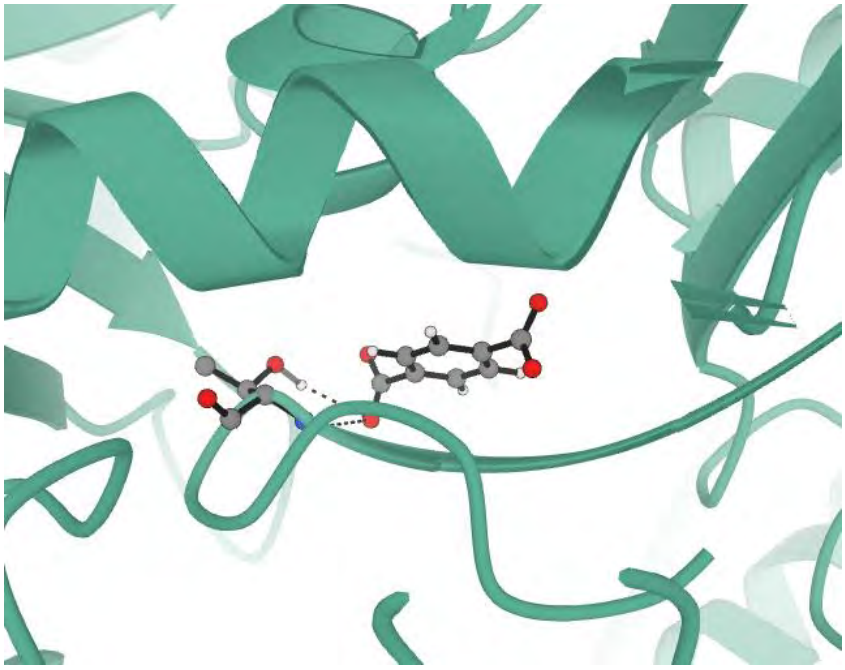
"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-3.024","0.8383528255522105"
"2","-2.975","4.824293804747557"
"3","-2.277","1.4984929247976668"
"4","-2.26","4.442767967983404"



New Substrate: Mutated 1T5D (F184→K) with TPA

Results

"Pose","Docking Score","Ligand Strain (kcal/mol)"
"1","-2.908","11.020323595198795"
"2","-2.908","11.09938980481319"
"3","-2.735","0.047063220023855955"
"4","-2.399","0.6952806368280746"



Code

No scripts used. Everything was performed using Rowan GUI.

Results Summary

Mutation	Summary of Results
Phe 184 → Thr	Slightly worse docking score (-3.2 to -2.9) but high ligand strain. Higher strain for both 4CB and TPA compared to unmutated and higher TPA strain than 4CB. Ligand seems to adopt the “wrong” “sideways” pose that causes this high strain like the Asn 311 mutations. Not a good mutation.
Phe 184 → Lys	Kind of an interesting case. Mutation on the original substrate only worsens the binding score *slightly,* and reduces the ligand strain. However, on TPA the ligand strain increases by a horrible amount! Ligand adopts again the “sideways” pose, and shows interaction between it and the Lysine residue. Not a good mutation, but what’s happening here?

Interpretation

Did not find any ‘good’ mutations.

Next Steps

Try more mutations on Phe184. Try mutations on other residues in the active site. Test more mutations with this workflow and come up with ideas on other different methods to test mutations.

2025-07-14 Co-folding of 1T5D with 4-chlorobenzoate and TPA

Contributors: Hilya

Objective

Compare co-folding of 1T5D with native substrate (4-chlorobenzoate) and TPA.

Background

On Co-Folding and Binding-Affinity Prediction with Boltz-2: Co-Folding and Binding-Affinity Prediction with Boltz-2 | Rowan, which would be a valuable tool in assessing ligand compatibility for this project moving forward.

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/)
Input Structure	4-chlorobenzoate TPA 1T5D
Key Parameters	pLDDT and ligand affinity
Script Location	None used.

Code

None used. All performed with Rowan GUI

Results Summary

- Looked into Rowan protein cofolding and submitted 1T5D with both (1) 4-chlorobenzoate and (2) TPA. pLDDT is good for both but ligand affinity decreases. Values and interpretations by DeepSeek:

Metric	Native Substrate	TPA (Our Substrate)	Interpretation
TM-Score (pTM)	0.930	0.925	Both structures are near-native (excellent). Minimal difference.

ipTM	0.985	0.961	Native substrate has a more accurate binding interface.
Confidence Score	0.969	0.963	Both predictions are highly reliable (no practical difference).
Average pLDDT	0.965	0.963	Near-identical residue-level confidence.
Affinity Probability	0.576	0.386	Native substrate is ~1.5× more likely to bind (TPA is borderline non-binder).
Predicted IC50	13.1 μM	357.3 μM	Native substrate binds 27× stronger than TPA.

Interpretation/Discussion

See result summary table.

Next Steps

- Make mutations (decide on what, or do all 19 for each residue?) and test them with co-folding.
- Suggestions from Amijai (ex-Baker Lab)
    - That *actually we shouldn't modify the active site….*
    - Suggested that we keep the active site intact but make the protein better by using MPNN (what Edu is already doing)
    - He suggested we take a look at a few tools, such as PLACER and Fleishman Lab tools. https://www.fleishmanlab.org/
    - He advised to try cofolding in AlphaFold3 but seems like it's not open source yet (only available for selected ligands).
        - From ChatGPT:
          AlphaFold 3 Server Access (Reality Check) As of now: AlphaFold 3 is not publicly open-sourced. There's no public web server where you can upload custom ligands directly to run full AF3 predictions. The official AF3 ligand-binding predictions were demonstrated with limited compound libraries, not custom SMILES or 3D files from users.

2025-07-14 Notes on 3TSY

Contributors: Sixten

Objective

Doublecheck 3TSY sequences before ordering for the genes for wet lab experiments.

Overview/Background

Realized I need a much better understanding of the enzyme I am working with (3TSY) and also the origin of the crystal structure used for docking. It appears that the specific gene coding for the crystal structure 3TSY is called At4CL1 (they exist in many different isoforms). Probably good to double check so that it is this exact gene we have ordered. I also wanted to look at modeling the solubility of the enzyme. A pipeline would be to model the FASTA in AlphaFold since the enzymes lack structural data.

Results/Discussion

- IMPORTANT TERMS OF SERVICE IN ALPHAFOLD 3:

Key things to know

1. AlphaFold Server is **only** available for non-commercial use by individuals and non-commercial organizations (universities, non-profit organizations and research institutes, educational and government bodies), or for journalism.
2. You **must not** use AlphaFold Server or its outputs:
   a. in connection with **any commercial activities, including research on behalf of commercial organizations;**
   b. in **any automated system that predicts the binding or interaction of the protein with ligands or peptides**, such as Glide or AutoDock; or
   c. to **train machine learning models** or related technology for **biomolecular structure prediction** similar to AlphaFold Server.
3. You **can publish, share and adapt** AlphaFold Server output in accordance with our terms, including the requirement to provide clear notice that ongoing use is subject to AlphaFold Server Output Terms of Use and of any modifications you make.

- This means we cannot do any docking with AlphaFold 3 output.
- Predicting the structure of the LCC enzyme using AlphaFold3. Should be able to use this output for solubility predictions. https://www.ncbi.nlm.nih.gov/protein/XHB18902.1?report=fasta
- Succesfully generated an LCC structure.
- Ok, final roadblock met. Apparently the workflow for solubility predictions in Rowan only takes 2D structures as input, meaning it is definitely not supposed to be used for protein solubility prediction.

- More on the understanding of the 4-coumaryl ligase. We might have messed up a bit. It seems like 3TSY protein is a fusion protein, i.e a combination of the 4-coumaryl ligase and a stilbene synthase. This is not good I think, but could explain the weird structure. I mean the enzyme should still have activity.
- Ideas for future modeling more connected to degradation:
    - Solubility of enzymes in ethylene glycol
- Looking up ways to do the solubility prediction
    - https://www.researchgate.net/publication/51029512_Recent_Advances_on_Aqueous_Solubility_Prediction
- Seems like it is fairly easy to make solubility predictions for water environments. Maybe hard for organic solvent like ethylene glycol.
- We could maybe do molecular dynamics to predict solubility
- Train our own model that can predict solubility of enzymes in different concentrations ethylene glycol? AI prize? (No probably not doable due to the need of large datasets. But would definitely be interesting and greatly contribute to enzymatic glycolysis).

Next Steps

- Things to talk about next meeting:
    - Coumarate enzyme: how do we proceed?
    - Solubility predictions
- Ask Edu about the possibility of using University's servers to do the solubility predictions. Would be great.

2025-06-15 Notes on questions for the protein design studio workshop

Contributors: Sixten

Objective

Think about questions to ask for the protein design studio workshop that some dry lab members will attend.

Overview/Background

We have performed dockings, but we want to have other methods to test proposed mutations.

Results/Discussion

- Questions for protein design studio:
  - Any tips on how to interpret the different dockings? Right now I am using mostly organic chemistry
  - Any other dockings that could be good to do? Control dockings? Other mutations?
  - The proposed mutation right now is Ile 303 → Ser. Is there any other ways to test and validate?
  - The protein design studio guy talked about how proteinMPNN does not check if the “biochemistry” of the protein. https://services.healthtech.dtu.dk/services/NetSolP-1.0/ possible to use this service to check if the generated sequences work in an E.coli environment? Maybe also ask DTU if they know anything about this.
  - Any “easy” way to predict solubility and stability in different ethylene glycol concentrations?
- Questions for DTU:
  - Ask about their solubility and stability service. Do they know anything about it? Is it possible to test in a different environment?

Next Steps

- Learn more about:
  - Placer
  - Check net charge of enzyme, if positive = bad, negative =? And why?
  - Can check how hard the PETases are to express
  - Fleishman institute, look for applied protein optimization
  - Interesting paper: Robust deep learning based protein sequence design using protein MPNN

- We need to find a better way to validate our mutations. We performed the cofolding function in Rowan and got good results! Is this enough???? The mutation improved results.
- Maybe input the mutated FASTA into alphafold 3 and see if it generates the same structure?
- Maybe do stability prediction and or ddG

2025-07-16 Alignment of co-folding results of 1T5D mutations - incorporating AlphaFold3 in screening

Contributors: Sixten

Objective

Quality control of co-folding results by comparing co-folded structures of 1T5D with native substrate and TPA with other references (e.g. crystal structures).

Background

If the co-folded structures are “correct” then they should align with structures generated with other AI methods and also with the crystal structures, so that alignment of these structures can be used as a screening or QC method for our pipeline.

What we want to test:

- The two cofolding structures from Rowan
    - mutated 1t5d Ile303 → Ser, one with TPA and one with 4-chlorobenzoate
- Structure of mutated 1t5d Ile303 → Ser generated from alphafold
- The crystal structure from uniprot website.

Methods and parameters

Component	*Description*
Software Used	AlphaFold3, Rowan, PyMOL
Input Structure	1T5D Mutation: Ile303toSer 1T5D sequence with mutation (for AlphaFold)
Key Parameters	pLDDT from co-folding, RMSD from alignments
Script Location	None

- Running some cofolding of the Ile 303 → Ser mutation in Rowan.

Cofolding with TPA:

Co-Folding Scores | Sequences

Info | Notes | JSON

Property	Value
Predicted TM-Score (pTM)	0.930
ipTM	0.964
Confidence Score	0.964
Average pLDDT	0.964
Affinity Probability	0.455
Predicted pIC_{50}	3.673
Predicted IC_{50}	212.3 µM

Cofolding with 4-chlorobenzoate:

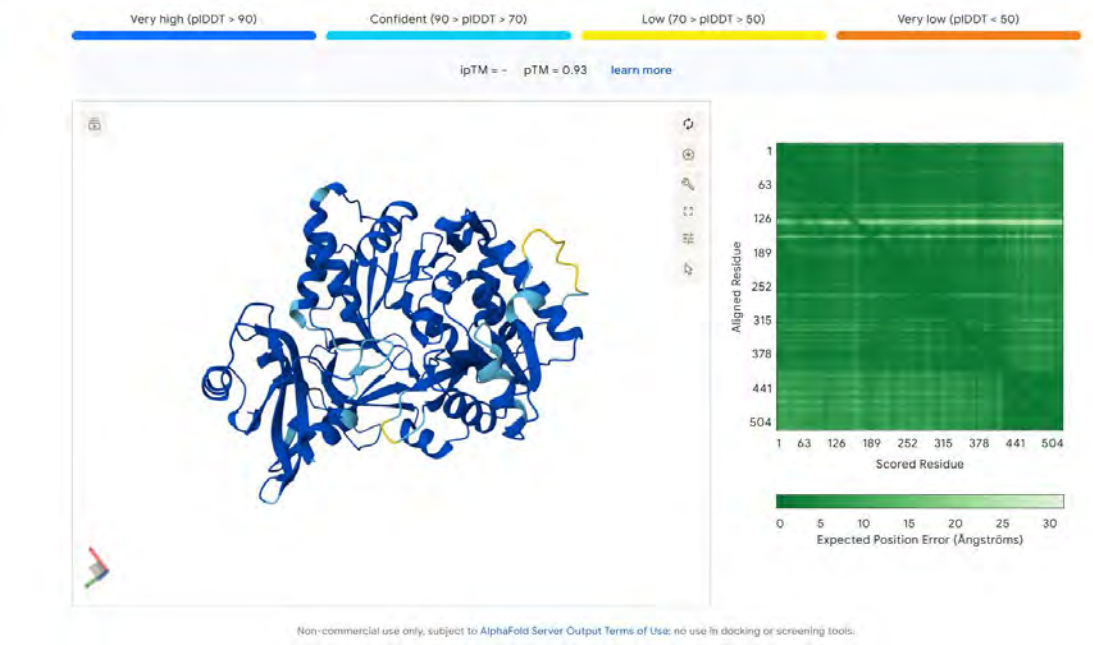
Co-Folding Scores | Sequences

Info | Notes | JSON

Property	Value
Predicted TM-Score (pTM)	0.928
ipTM	0.983
Confidence Score	0.968
Average pLDDT	0.964
Affinity Probability	0.544
Predicted pIC_{50}	4.796
Predicted IC_{50}	16.0 µM

- Running the mutated structures in alphafold to see how the predicted structures match up with the ones in Rowan and with the expected structure.

Comparisons of the cofolding and alphafold structure in pyMOL:



- It appears that when doing the cofolding in Rand alphafold folding the coordinates of the atoms are not consistent even if the structures are the same. Not weird in itself since the cofolding and alphafold folding uses different seeds each time. Using the align function in pymol.
- Results from alignment from the two cofolding results from Rowan
    - Match: assigning 505 x 505 pairwise scores.
    - MatchAlign: aligning residues (505 vs 505)...
    - MatchAlign: score 2541.000
    - ExecutiveAlign: 3828 atoms aligned.
    - ExecutiveRMS: 165 atoms rejected during cycle 1 (RMSD=0.72).
    - ExecutiveRMS: 277 atoms rejected during cycle 2 (RMSD=0.45).
    - ExecutiveRMS: 216 atoms rejected during cycle 3 (RMSD=0.33).
    - ExecutiveRMS: 182 atoms rejected during cycle 4 (RMSD=0.26).
    - ExecutiveRMS: 139 atoms rejected during cycle 5 (RMSD=0.22).
    - Executive: RMSD = 0.196 (2849 to 2849 atoms)
- Really good. The structures are nearly perfectly identical.
- Performing the alignment for more structures. These include: The two cofolding structures from Rowan (mutated 1t5d Ile303 → Ser, one with TPA and one with 4-chlorobenzoate); the structure of mutated 1t5d Ile303 → Ser generated from alphafold and also the crystal structure from uniprot website. Using the crystal structure as base structure. Results:
    - PyMOL>align alphafold and polymer.protein, 1t5d and polymer.protein
      Match: read scoring matrix.
      Match: assigning 504 x 502 pairwise scores.
      MatchAlign: aligning residues (504 vs 502)...
      MatchAlign: score 2508.500
      ExecutiveAlign: 3781 atoms aligned.
      ExecutiveRMS: 247 atoms rejected during cycle 1 (RMSD=2.23).

ExecutiveRMS: 262 atoms rejected during cycle 2 (RMSD=1.55).
ExecutiveRMS: 271 atoms rejected during cycle 3 (RMSD=1.16).
ExecutiveRMS: 228 atoms rejected during cycle 4 (RMSD=0.86).
ExecutiveRMS: 151 atoms rejected during cycle 5 (RMSD=0.65).
Executive: RMSD = 0.559 (2622 to 2622 atoms)

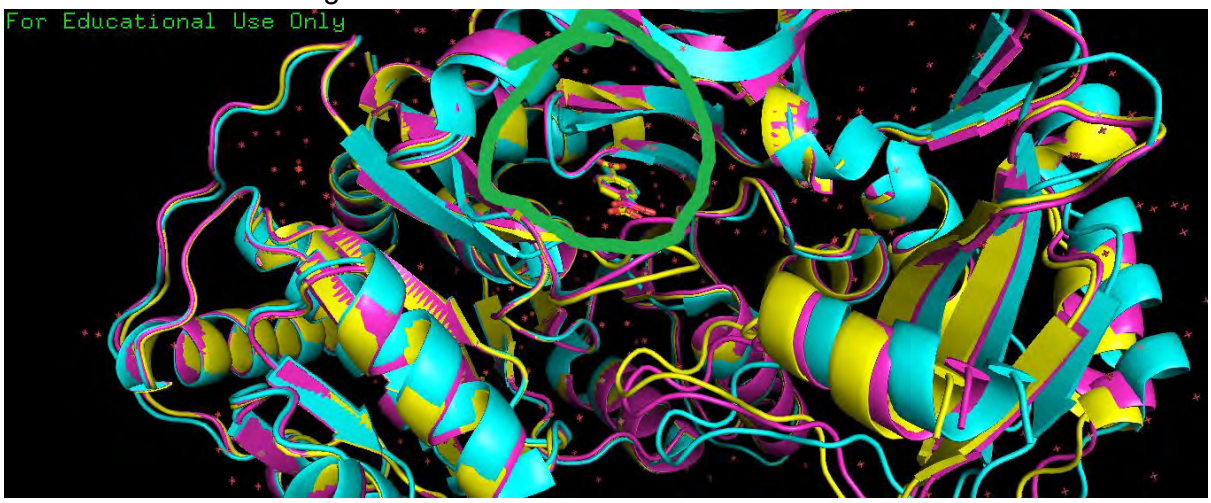
- PyMOL>align 4CB and polymer.protein, 1t5d and polymer.protein

Match: read scoring matrix.
Match: assigning 504 x 502 pairwise scores.
MatchAlign: aligning residues (504 vs 502)...
MatchAlign: score 2508.500
ExecutiveAlign: 3781 atoms aligned.
ExecutiveRMS: 186 atoms rejected during cycle 1 (RMSD=1.52).
ExecutiveRMS: 265 atoms rejected during cycle 2 (RMSD=1.01).
ExecutiveRMS: 227 atoms rejected during cycle 3 (RMSD=0.78).
ExecutiveRMS: 202 atoms rejected during cycle 4 (RMSD=0.63).
ExecutiveRMS: 144 atoms rejected during cycle 5 (RMSD=0.52).
Executive: RMSD = 0.461 (2757 to 2757 atoms)

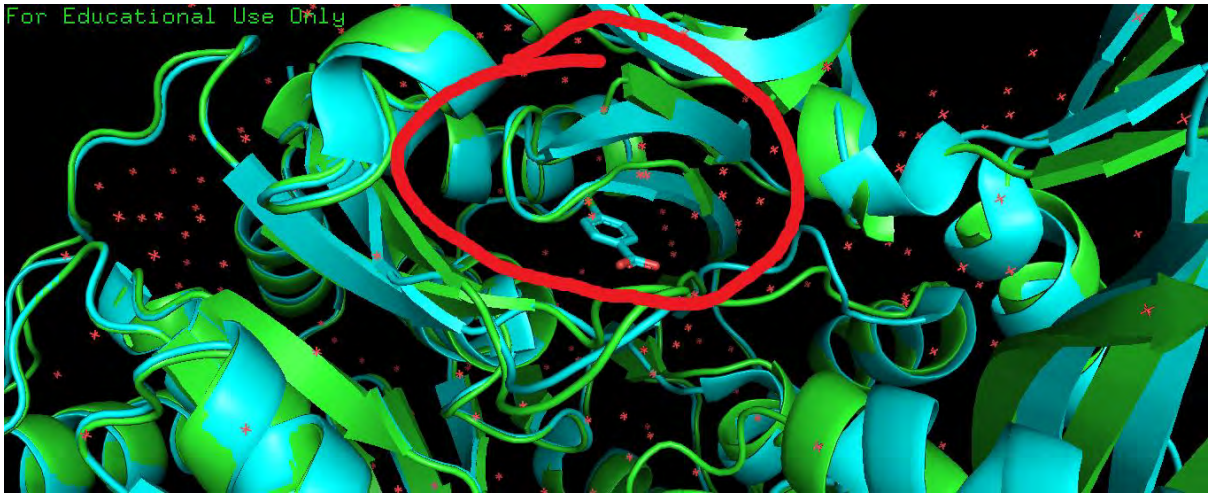
- PyMOL>align TPA and polymer.protein, 1t5d and polymer.protein

Match: read scoring matrix.
Match: assigning 504 x 502 pairwise scores.
MatchAlign: aligning residues (504 vs 502)...
MatchAlign: score 2508.500
ExecutiveAlign: 3781 atoms aligned.
ExecutiveRMS: 219 atoms rejected during cycle 1 (RMSD=1.71).
ExecutiveRMS: 289 atoms rejected during cycle 2 (RMSD=1.16).
ExecutiveRMS: 224 atoms rejected during cycle 3 (RMSD=0.83).
ExecutiveRMS: 159 atoms rejected during cycle 4 (RMSD=0.62).
ExecutiveRMS: 116 atoms rejected during cycle 5 (RMSD=0.53).
Executive: RMSD = 0.484 (2774 to 2774 atoms)

Crystal structure (cyan), cofolded with 4-chlorobenzoate (pink), cofolded with TPA (yellow). The active site is circled green.



Crystal structure (cyan), alphafold structure (green). The active site is circled red.



Code

None.

Results Summary

- **PyMOL>align alphafold and polymer.protein, 1t5d and polymer.protein**
  Executive: RMSD = 0.559 (2622 to 2622 atoms)
- **PyMOL>align 4CB and polymer.protein, 1t5d and polymer.protein**
  Executive: RMSD = 0.461 (2757 to 2757 atoms)
- **PyMOL>align TPA and polymer.protein, 1t5d and polymer.protein**
  Executive: RMSD = 0.484 (2774 to 2774 atoms)

Interpretation

- Since all RMSD are below 1 the alignments are good. It seems like the cofolding structures from Rowan are better aligned with the active site of the crystal structure (see below). Maybe this has to do with the cofolding having a ligand present.

Next Steps

- We think that this could be a good screening layer to our pipeline and future work will incorporate comparison with AlphaFold structures.

2025-07-17 Investigating 4-coumarate ligases 3TSY and 5BST, identifying the active site?

Contributors: Sixten

Objective

Determine the active site for 4-coumarate ligases. We want to see if 3TSY has the same active site as 5BST.

Background

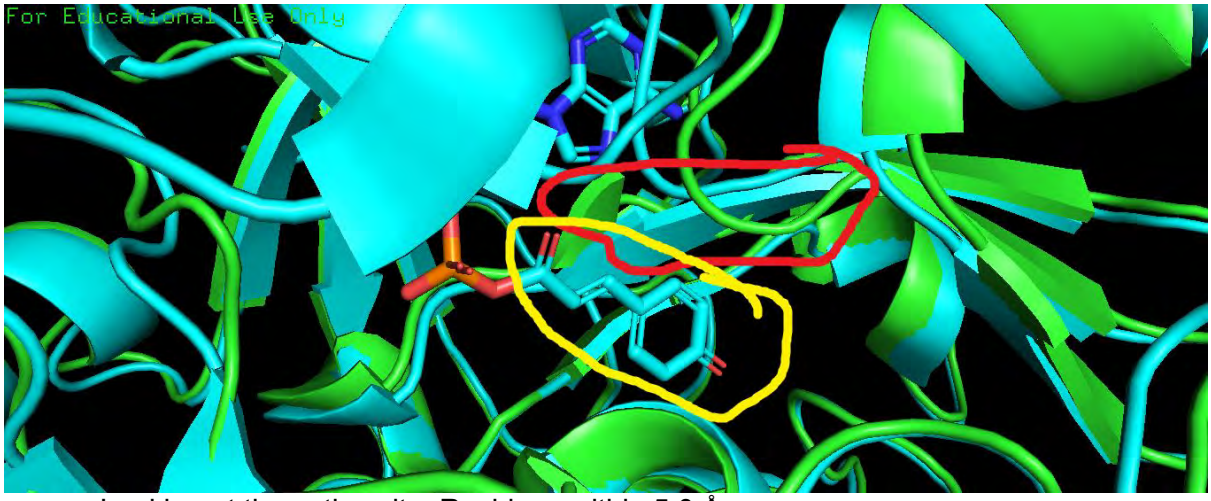
Starting to investigate the Coumarate for real. Since we have ordered a fusion protein (whoops) we need to do some confirmation that the active site of the fusion protein aligns with the active site of original coumarate ligase. To do this I am looking at a different isoform (a bit sus but could maybe work) that already has a docked molecule. This will hopefully help determine the active site.

Methods and parameters

Component	*Description*
Software Used	pyMOL
Input Structure	3TSY, 5BST Ligand(s): 4-coumarate
Key Parameters	RMSD
Script Location	No scripts used.

- Aligned the structures in pymol:
    - PyMOL>align 5bst and polymer.protein, pdb3tsy and polymer.protein
    - Match: read scoring matrix.
    - Match: assigning 542 x 828 pairwise scores.
    - MatchAlign: aligning residues (542 vs 828)...
    - MatchAlign: score 1653.000
    - ExecutiveAlign: 3190 atoms aligned.
    - ExecutiveRMS: 103 atoms rejected during cycle 1 (RMSD=8.07).
    - ExecutiveRMS: 144 atoms rejected during cycle 2 (RMSD=1.66).
    - ExecutiveRMS: 196 atoms rejected during cycle 3 (RMSD=1.07).
    - ExecutiveRMS: 160 atoms rejected during cycle 4 (RMSD=0.85).
    - ExecutiveRMS: 124 atoms rejected during cycle 5 (RMSD=0.73).
    - Executive: RMSD = 0.654 (2463 to 2463 atoms)
- Looks alright, RMSD is ok.

Alignment (circled red is biggest difference and circled yellow is 4-coumarate. Green is 3TSY and cyan is 5BST):



- Looking at the active site. Residues within 5.0 Å are:

For 3TSY:

Residue nr	Residue name
257	Gly
281	His
282	Ile
283	Tyr
287	Ser
350	Lys
352	Gly
353	Ala
376	Gly
377	Tyr
378	Gly
379	Met
384	Pro
385	Val
388	Met

For 5BST:

213	Gln
237	His
238	Ile

239	Tyr
243	Ser
306	Met
308	Gly
309	Ala
332	Gly
333	Tyr
334	Gly
335	Met
336	Thr
340	Pro
341	Val
344	Met
441	Lys
444	Gly

Code

No scripts used.

Results Summary

- So a summary of the workflow (now that we have done it for 1t5d) is:
    - 1. Align the isoform 5BST and the fusion protein 3TSY to see how active site matches.
    - 2. Analyse the active site based on the isoform 5BST in pymol. Check for residues within 5.0 Å.
    - 3. Prepare a pdb file of the coumarate ligase part of the fusion protein (3TSY) to continue modeling with.
    - 4. Check if the residues obtained in step 2 seem to make sense when highlighted in the pdb file from step 3.
    - 5. Begin docking in Rowan with the original substrate 4-coumarate and try to successfully dock it in the active site. If successful, dock TPA and then move on to point mutations. If not, a different approach has to be explored.

    - Perform the point mutations in pymol of residues part of the active site. Use rational design to determine potential candidates.
    - Dock the mutated protein in Rowan. Dock TPA and 4-coumarate. If docking results in better binding affinity, move on to validation of the mutation.
    - To validate:
        - Perform cofolding in Rowan with both TPA and 4-coumarate, with both original form and mutated form.
        - Check folding of structure in alphafold. Do this with both original form and mutated form.
        - Align the modeled structures and see if they are consistent.

- Executive: RMSD (3TSY and 5BST) = 0.654 (2463 to 2463 atoms)

Interpretation

Seems that the active site is still the same in the fusion protein.

Next Steps

We can proceed to test 3TSY with wet lab

2025-07-14 Docking coumarate to 3TSY in Rowan

Contributors:

Objective

Perform a docking in Rowan to see if 4-coumarate binds in a promising way to 3TSY.

Background

(see entry 2025-07-18) Since we have ordered 3TSY which is apparently a fusion protein (whoops) we need to do some checks that the substrate binds in the correct way to 3TSY.

Methods and parameters

Component	*Description*
Software Used	pyMOL, Rowan, AlphaFold
Input Structure	3TSY 4-coumarate coumaryl-AMP
Key Parameters	Substrate position and orientation in the protein
Script Location	Extract box

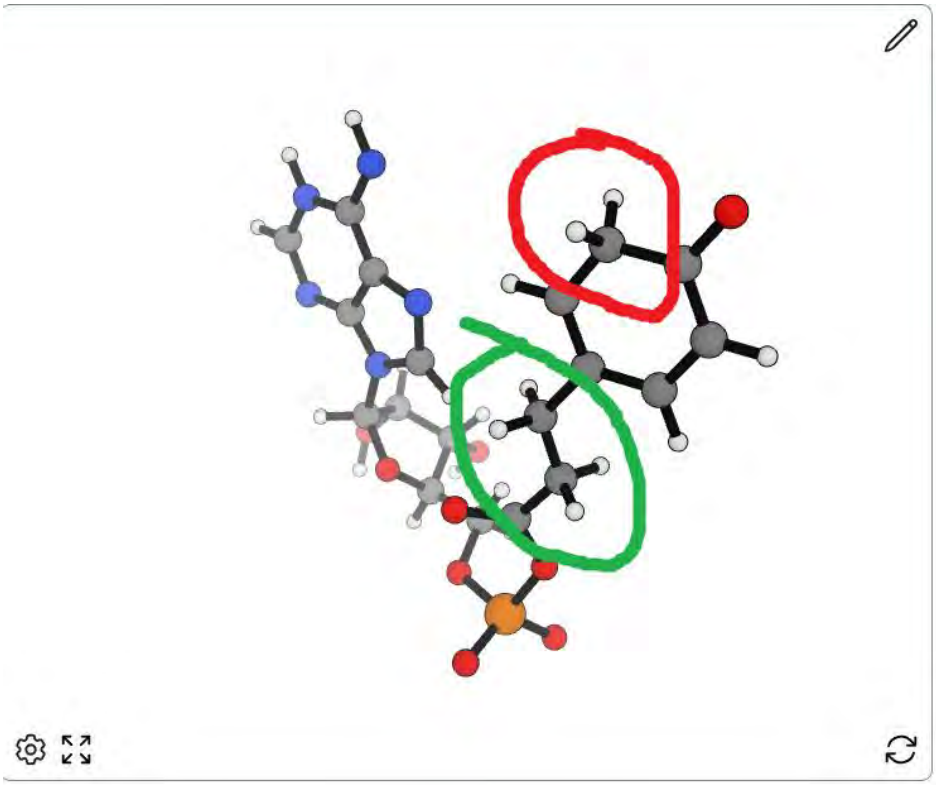
- Created a pdb file with only the coumarate ligase part of the 3TSY protein.
- When highlighting the proposed binding site in pymol the following box was obtained:

```
Bounding Box:
  Min coords: [70.47799682617188, -115.41500091552734, -50.39799880981445]
  Max coords: [88.25199890136719, -96.00299835205078, -31.857999801635742]

Use these for docking:
  Center     = [79.36499786376953, -105.70899963378906, -41.1279993057251]
  Box size   = [17.774002075195312, 19.412002563476562, 18.53999900817871]
```

- Will perform a docking in Rowan to see if 4-coumarate binds in a promising way.
- Running a AlphaFold prediction just to see if the predicted cut coumarate ligase folds in a good way.
- Docking results were not that good. None of the docked ligand positions match the docked ligand of 5BST. I am thinking that making the box smaller is not the way to go but not sure.
- Will try to dock the whole coumaryl-AMP molecule to see if any difference.

- So, when investigating docking with coumaryl_AMP I stumbled upon something weird. I downloaded the coumaryl_AMP ligand from the docked 5BST file but when looking at it in Rowan it appears to be the wrong structure:



- As you can see the benzene ring of coumarate part only has two double bonds for some reason (highlighted red, this is the case because one carbon otherwise has more than four bonds). There should also be a double bond where it is highlighted green, but there cannot be since there are two many hydrogens. Theory first was that something weird happened when creating the sfd file but it appears the docked molecule form 5BST pdb file also has a two double bond benzene ring.
- Right now struggling alot with the coumaroyl-AMP sigh.
- Got the idea to make a script that does mutations for you in pymol. This would make the process of finding good mutations alot faster.
- Testing it.
- It works

Code

- For extract box script see entry 2025/06/16

Language: Python

Script:

```
# Python
# This program prints Hello, world!
```

```
print('Hello, world!')
```

Results Summary

- None of the docked ligand positions match the docked ligand of 5BST.
- For docking, making the box smaller might not be the solution. → other issue rather than definition of ligand position

Interpretation

- Could it be that the 3TSY binds a different state of substrate than 5BST, that is not coumaryl AMP or coumarate?

Next Steps

- The automated mutation script works, need automation of docking now. Now just have to get the API on Rowans website to work. Emailed a guy working on Rowan and he said they will launch a new API this week, making it possible to automate many of the docking steps.

2025-07-21 Co-folding coumarate with 3TSY in Rowan

Contributors: Sixten

Objective

Cofold in Rowan to see if 4-coumarate binds in a promising way to 3TSY. Compare co-folded structure with other structures of 4-coumarate CoA ligase, e.g. 5BST.

Background

(see entry 2025-07-18) Since we have ordered 3TSY which is apparently a fusion protein (whoops) we need to do some checks that the substrate binds in the correct way to 3TSY.

Methods and parameters

Component	*Description*
Software Used	pyMOL, Rowan
Input Structure	3TSY, 5 BST 4-coumarate
Key Parameters	Substrate position and orientation in the protein
Script Location	Extract box

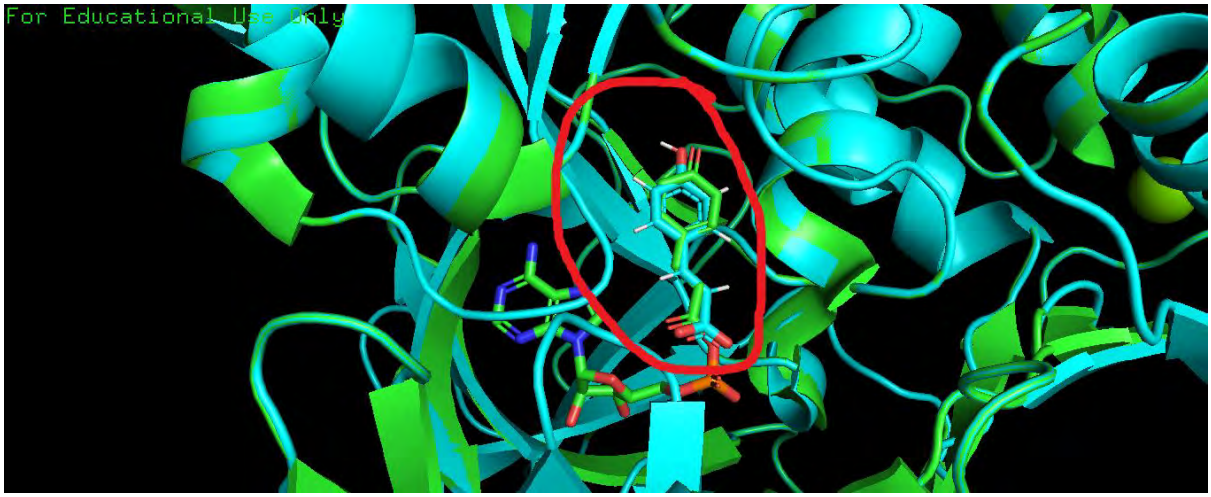
- Regarding the coumarate-AMP we discussed the structure and figured it might still be ok. However, converting the smile from pubchem to 3D does not result in a reasonable structure.
- Doing docking on the 5BST structure to see if docking works better there. Doing one blind docking and then one with a box (doing the docking with only 4-coumarate).
- Nevermind, not doing a blind docking. Seems unnecessary. Extracted a box

```
Bounding Box:
  Min coords: [-10.795999526977539, 18.756999969482422, 8.355999946594238]
  Max coords: [15.32699966430664, 40.73099899291992, 26.829999923706055]

Use these for docking:
  Center     = [2.265500068664551, 29.743999481201172, 17.592999935150146]
  Box size   = [26.12299919128418, 21.9739990234375, 18.473999977111816]
```

- Notice that the box is very different compared to the one for 3TSY due to alignment changing the coordinates of the proteins. But honestly, when comparing them, it still looks a bit wierd
- Docking Successful! The docked ligand overlaps with the docked coumarate-AMP in 5BST

Picture of 5BST (green) and 3TSY (coumarate ligase part) (cyan)with their respective ligands



- Now I just have to figure out how to proceed.
- I have remade the alignment in pymol to get coordinates that matches the 5bst coordinates, this to exclude any potential errors related to the bounding box.
- Investigating the coumarate a bit more now. So I had just fixed the coordinates for the 3TSY part. Performing docking again with native substrate to see if it is better.
- It does not dock in the same position. This is a pretty big problem. Either we put effort into making it dock correctly or we keep on investigating the 5bst one and assume the same mutations will be sufficient for 3tsy (very sus). In my opinion we should put effort into trying to make it dock properly.
    - I have two ideas:
        - Use the cofolding function in Rowan to see if it yields any different result.
        - Download the coumarate part of the 3TSY protein (i.e not 5bst but a different one) and see if it docks better.
- Going to proceed with both options
    - If cofolding works it suggests that there is a problem with the crystal structure of 3TSY. This would not be that weird since that crystal structure is a fusion protein (cough cough) so if that is the case we would have to look at other ways to get the structure of 4CL1 ARATH.

Cofolding results:

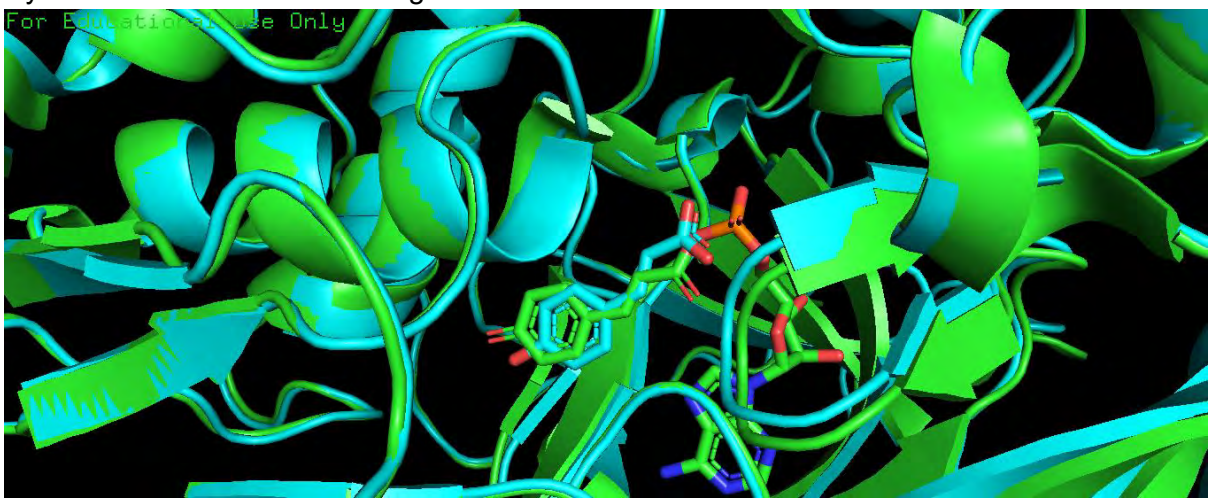
Co-Folding Scores | Sequences

Info | Notes | JSON

Property	Value
Predicted TM-Score (pTM)	0.914
ipTM	0.971
Confidence Score	0.939
Average pLDDT	0.931
Affinity Probability	0.518
Predicted pIC_{50}	4.077
Predicted IC_{50}	83.8 µM

- Looks pretty good. Alignment good as well:

Cyan is cofolded structure and green is 5bst:



- Conclusion: There is likely something weird with the structure of 3TSY that messes with the docking. How to proceed: Not sure. There is no crystal structure of only the 4CL1 ARATH gene (mouse ear cress). Could perhaps take it from a different species. Other way is to perform docking with the predicted structure (the cofolded structure).

Code

For script extracting box see entry 2025/06/16

Results Summary

- Ligand does not dock in the same position in 3TSY and 5BST!
- However, co-folding 3TSY works, i.e., good alignment of cofolded 3TSY bound to coumarate with 5BST

Interpretation

There is likely something weird with the structure of 3TSY that messes with the docking. How to proceed: Not sure. There is no crystal structure of only the 4CL1 ARATH gene (mouse ear cress). Could perhaps take it from a different species. Other way is to perform docking with the predicted structure (the cofolded structure).

Next Steps

- Decide how to move forward with this information. ProteinMPNN should use 5BST.

2025-07-21 Co-folding 1T5D MPNN structures

Contributors: Sixten

Objective

See how our generated proteinMPNN sequences co-fold with the substrates, before performing mutations.

Background

ProteinMPNN team has successfully finished some sequences for 1T5D. We want to try see how the substrate binds.

Methods and parameters

Component	*Description*
Software Used	pyMOL, Rowan
Input Structure	Given below 1T5D pMPNN structures A to E Mutated A Ile303toSer Ligands: TPA, 4-chlorobenzoate
Key Parameters	Substrate position and orientation in the protein
Script Location	None

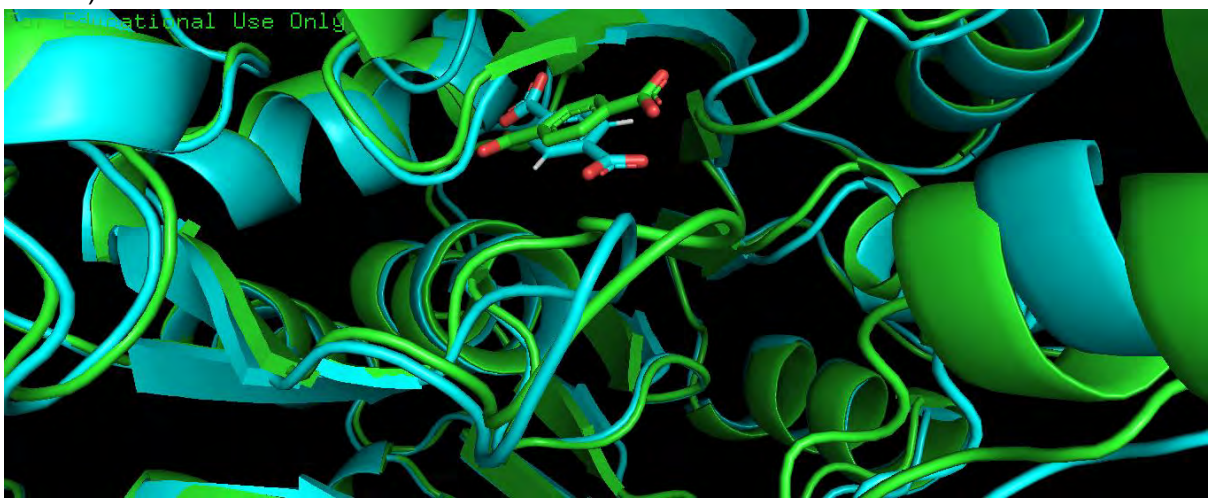
- Obtained the proteinMPNN sequences from the proteinMPNN team! Will check how it cofolds in Rowan, make a point mutation (Ile 303 → Ser). Should also make the dockings.
- Results from cofolding with TPA (we should do with 4-chlorobenzoate as well but waiting a little bit) for the first structure (MRTVVDLILDAAQKHPDSPALIDPEKGIRLTHAQFLAAVRAVAARLAALGLRPGQRVAIVAYNSI DTVLVILALLTLGAVPFLIDPRLPAEEIAALVRHAECAAVVVSPGDAVREAVEASGSGARVIPLD SLVRNGVPTIFGPPVPLPARTPDAPAVIFLDAXXTGPPRGVVIPHRAAEPRVSFMQGWCGLKPGT DNVVLGLMPLYHVVGFFGQLITALALGGTYVIVEKFDAKKALDLIQREKVTAVFATPEQLKQLAE AAADPNSTLDLSSLKHVIFAGTEMTDEDLALIHKNLPGTLVNIYGTTEAMNFAYRQNPTSGTVLT PGPGAEVRVVKIGGGVDEVVAKGEVGELIVRAGPAAATGYLNQPEATAARFVDGWYRTGDLARWT DDGSIELLGKISDAIISKGRLIFPQKVERVLSKAPGVAEVEVIGLPDSNGTEEVTAVVVPEPGKT LSEEALMAYLKASDLPEYERPKRFFIVDSLPRDHLNRIRREALREMIAA)

Info Notes JSON

Property	Value
Predicted TM-Score (pTM)	0.898
ipTM	0.933
Confidence Score	0.937
Average pLDDT	0.937
Affinity Probability	0.478
Predicted pIC_{50}	3.920
Predicted IC_{50}	120.2 µM

- Binding affinity for TPA increased, and the other scores are pretty good! However, when looking at it in pymol it seems the cofolding does not result in the exact binding position, even if it is pretty close. This is probably not good.

Cofolded ProteinMPNN1 sequence (green) and first successful docking of TPA to native 1t5d (cyan)(should be crystal structure but they are very similar so for conclusion it does not matter):



- Conclusion. The ones that align well with the crystal structure should be prioritized. Now performing the co-folding with the rest to see if it matches. Waiting for a talk with Edu.
- Doing cofolding with all of the structure Edu generated.
- Other sequences to test:

MRTVVDLILDAAQKHPDSPALIDPEKGIRLTHAQFLAAVRAVAARLAALGLRPGQRVAIVAYNSI
DTVLVILALLTLGAVPFLIDPRLPAEEIAALVRHAECAAVVVSPGDAVREAVEASGSGARVIPLD
SLVRNGVPTIFGPPVPLPARTPDAPAVIFLDAXXTGPPRGVVIPHRAAEPRVSFMQGWCGLKPGT
DNVVLGLMPLYHVVGFFGQLITALALGGTYVIVEKFDAKKALDLIQREKVTAVFATPEQLKQLAE
AAADPNSTLDLSSLKHVIFAGTEMTDEDLALIHKNLPGTLVNIYGTTEAMNFAYRQNPTSGTVLT
PGPGAEVRVVKIGGGVDEVVAKGEVGELIVRAGPAAATGYLNQPEATAARFVDGWYRTGDLARWT

DDGSIELLGKISDAIISKGRLIFPQKVERVLSKAPGVAEVEVIGLPDSNGTEEVTAVVVPEPGKT
LSEEALMAYLKASDLPEYERPKRFFIVDSLPRDHLNRIRREALREMIAA

MKTVIDLVLEAAKKYPDQPALIDPEKGIRLTHAEFLEAVRAVARALAAMGLQPGQRVAVIAPNSL
DTVLVILAILYLGCIPMLIDPSLPPEVIAELVKHDEAAAVVYAPGDEVAKAIRDSGSGARVIPLS
SLVVDGVPRVFGPEVPLPERLPDAPYVIFYTNXXKGLPLGAVIPHRAAESRVLFMSTQVGLTPGT
DNVVLGLMPLYHVVGFFAVLVQALALGGTYVVVEKFDADKVLKLIQDEKVTSLFATPAQLAELAA
AAADPNSTLDLSSLKHVTFAGATMTDEDLARVRANLPGTLVNIYGTTEAMNSLYAVNPTRGTVMR
PGAFSEVRVIKPGGGVNEVVEPGEVGELIVKAGDSAATGYLNNPEATAKRFIDGWYRTGDLAVYT
PDGTIEILGRLEDAIISNGRLIFPQRVEAVLSKAPGVKDVVVVGLPNDNGTQDVTAVVVPEEGKT
LSEEALLAFCAASDLPAYEIPKKVFIVDSLPKNHLNKIMKEELKKMIES

- It appears the sequence I cofolded earlier is different (have to ask Edu about this) than the one Edu sent later (that had gone through chai).

- Doing the cofolds with the new sequences he sent later. Naming them A, B, C, D, E. Just to write down which sequence are named what:

pMPNN sequences to test, A to E

,file_ID,seq,pLDDT,RMSD,RMSD_norm,pLDDT_norm,score
7,msa_14-07/organised_chai_structures/1t5d.pdb_1st-shell_2nd-shell_treshold-95_seq_7.cif,"('MKTVIDLVLEAAKKYPDQPALIDPEKGIRLTHAEFLEAVRAVARALAAMGLQPGQRVAVIAPNSLDTVLVILAILYLGCIPMLIDPSLPPEVIAELVKHDEAAAVVYAPGDEVAKAIRDSGSGARVIPLSSLVVDGVPRVFGPEVPLPERLPDAPYVIFYTN', 162)",90.05377521231422,1.162112982869668,1.0,0.771888457730567,0.9087553830922268
23,msa_14-07/organised_chai_structures/1t5d.pdb_1st-shell_2nd-shell_treshold-95_seq_3.cif,"('MKTVIDMVTEATKKYPDQPALVDPEKGISLTHAQFDAAVAAVAARLAAMGLRPGQRVGVVSRNSLDTVLAILAILRLGAVPALMDPRLPAAVIAELVKRDKLAAVVLSPGDEVAAAIAASGSGARVIRLSDLVVDGVPTVSGAPVPTPRREPEEPAVIFYTA', 162)",90.35812610441766,1.3068841437363514,0.9046145879245538,0.8413117041520837,0.8792934344155657
11,msa_14-07/organised_chai_structures/1t5d.pdb_1st-shell_2nd-shell_seq_5.cif,"('MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDPRLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTG', 162)",90.31692084221748,1.3546320311789382,0.8731549250180911,0.8319126744603488,0.8566580247949942
38,msa_14-07/organised_chai_structures/1t5d.pdb_1st-shell_2nd-shell_seq_2.cif,"('MKTVVEMVLEAAEKYPDQPALIDPDKGIELTHAEFLDAVDAVAARLAAMGLKPGERVAVISPNSLDTVIVILAILRLGAIPVLIDPRLPAEVIAELVKRDKCKAVVYDEGKEVKEAIEKSGSGAKIIPLEDLVRDGVPTVFGPPVPLPERRPDEPYVIFYTS', 162)",90.01801553040274,1.3444729028467575,0.8798484725208617,0.7637315793585714,0.8334017152559455
5,msa_14-07/organised_chai_structures/1t5d.pdb_1st-shell_2nd-shell_treshold-95_seq_9.cif,"('MKTVIDMVTEAAAKHPDHPALVDPAAGIALTHAEFMAAVRAVARRLAALGLKPGERVAVIAPNSLDTVLVILALLRLGAIPVLIDPRLPAELIAELVKHNRCRAAVISPGEEVEEA

IRESGSGARVIRLEDLVRDGVPDAFGPEIPEPEFRPDDPYVIFYTG',
162)",89.23461045442606,1.2405728680606098,0.948305115220966,0.585034801893859
1,0.8029969898901232

A = RED
B = GREEN
C = BLUE
D = PURPLE
E = PINK

- Realized the sequences given were wrong. Will come back to this later.

Code

None

Results Summary

Seq	Results (pTM, ipTM, confidence, avg pLDDT, affinity probability, predicted PIC50, predicted IC50)
A	
B	
C	
D	
E	

(No results, used the wrong sequences).

Interpretation

Co-fold performs well, but need to recheck whether I was using the right sequences…

Next Steps

- Re-check with Edu which sequences are correct
- Perform mutations for the MPNN sequences and test them according to our workflow

2025-07-22 Evaluating ddG of 1T5D mutants using FoldX

Contributors: Hilya

Objective

Calculation of ddG of 1T5D mutants and evaluate mutant by stability.

Background

One of the effects of mutations on an enzyme is the impact to the stability of the enzyme. Stability can be assessed y calculating the free energy difference (ΔG) of unfolding. A mutation can stabilising or destabilising depending on the difference of the ΔG unfolding of the mutant and the wild type.

Reference: Bromberg, Y., & Rost, B. (2009). Correlating protein function and stability through the analysis of single amino acid substitutions. BMC Bioinformatics, 10(S8). https://doi.org/10.1186/1471-2105-10-s8-s8

Methods and parameters

Component	*Description*
Software Used	FoldX
Input Structure	1T5D
Key Parameters	
Script Location	

- Downloaded FoldX 5.1 on linux PC.

```
chmod +x foldx
```

- In my case the executable is called foldx_50253112…

```
chmod +x foldx_20253112
```

- Download 1t5d into the FoldX 5.1 directory
- Cleaned 1t5d structure with repair:

```
./foldx --command=RepairPDB --pdb=your_protein.pdb
```

- Made a mutation file (in this case, individual_list)
- Follow the syntax here: [mutant-file | FoldX]
- I made a list for an Ile 303 to Ser mutation.
- Don't forget to create the output folder first (here it's results)

```
./foldx --command=BuildModel --pdb=your_protein_Repair.pdb  --
mutant-file=individual_list.txt  --output-dir=results  --numberOfRuns=5
```

- The ddG can be shown in the Dif_1t5d_Repair file.
- Example of output:

```
FoldX 5.1 (2011)
by the FoldX Consortium
Jesper Borg, Frederic Rousseau, Joost Schymkowitz,
Luis Serrano and Francois Stricher
-------------------------------------------------------

PDB file analysed: batch
Output type: BuildModel
Pdb   total energy    Backbone Hbond  Sidechain Hbond Van der Waals
     Electrostatics  Solvation Polar Solvation Hydrophobic Van der
Waals clashes   entropy sidechain     entropy mainchain
     sloop_entropy   mloop_entropy   cis_bond   torsional clash
     backbone clash  helix dipole    water bridge    disulfide
     electrostatic kon     partial covalent bonds     energy
Ionisation Entropy Complex
1t5d_Repair_1_0.pdb   3.43099    0.0778749  0.00932837 1.69341    0
     -0.985338  3.19195    -0.0430048 -0.98266   0.480948   0     0
     0     -0.0115136 -0.0570682 0     0     0     0     0     0     0
1t5d_Repair_1_1.pdb   0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0
1t5d_Repair_1_2.pdb   0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0
1t5d_Repair_1_3.pdb   0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0
1t5d_Repair_1_4.pdb   0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0
```

- So the 303 Ile to Ser mutation has a +ddG of 3.43.

Code

See commands above.

Results Summary

Successfully calculated ddG of a mutation in 1T5D. 303 Ile to Ser mutation has a +ddG of 3.43.

Interpretation

Remember that this method doesn’t take into account ligands and think about the implications to our project.

Next Steps

Come up with automated scripts to make mutations and test the ddG using FoldX.

2025-07-22 Docking of substrates to 1T5D MPNN structures

Contributors: Sixten

Objective

See how substrates binds with our generated proteinMPNN sequences with the substrates, before performing mutations.

Background

ProteinMPNN team has successfully finished some sequences for 1T5D. We want to try see how the substrate binds. (See entry 2027-07-21 (2)) Previously we performed co-folding with the wrong MPNN structures. Will do a docking now, hopefully with the correct sequences.

Methods and parameters

Component	*Description*
Software Used	pyMOL, Rowan
Input Structure	Given below 1T5D pMPNN structures A to E, 3D structures Ligands: TPA, 4-chlorobenzoate
Key Parameters	Substrate position and orientation in the protein
Script Location	

A = RED
B = GREEN
C = BLUE
D = PURPLE
E = PINK

MKTVIDLVLEAAKKYPDQPALIDPEKGIRLTHAEFLEAVRAVARALAAMGLQPGQRVAVIAPNSLDTVLVI
LAILYLGCIPMLIDPSLPPEVIAELVKHDEAAAVVYAPGDEVAKAIRDSGSGARVIPLSSLVVDGVPRVFG
PEVPLPERLPDAPYVIFYTNXXKGLPLGAVIPHRAAESRVLFMSTQVGLTPGTDNVVLGLMPLYHVVGFFA
VLVQALALGGTYVVVEKFDADKVLKLIQDEKVTSLFATPAQLAELAAAAADPNSTLDLSSLKHVTFAGATM
TDEDLARVRANLPGTLVNIYGTTEAMNSLYAVNPTRGTVMRPGAFSEVRVIKPGGGVNEVVEPGEVGELIV
KAGDSAATGYLNNPEATAKRFIDGWYRTGDLAVYTPDGTIEILGRLEDAIISNGRLIFPQRVEAVLSKAPG

VKDVVVVGLPNDNGTQDVTAVVVPEEGKTLSEEALLAFCAASDLPAYEIPKKVFIVDSLPKNHLNKIMKEE
LKKMIES

MKTVIDMVTEATKKYPDQPALVDPEKGISLTHAQFDAAVAAVAARLAAMGLRPGQRVGVVSRNSLDTVLAI
LAILRLGAVPALMDPRLPAAVIAELVKRDKLAAVVLSPGDEVAAAIAASGSGARVIRLSDLVVDGVPTVSG
APVPTPRREPEEPAVIFYTAXXDGLPKGAVIPHRAAESRVLFMSTQVGLTPGPDNVVLGLMPLYHVVGFFA
VLISALALGGTYVIVETFDAQRVLQLIQDEKVTSLFATPAQLKALAEAAADPNSTLDLSSLKYVTFAGATM
TRADLAAVRANLPGRLVNIYGTTEAMNSLYKEEPEVGTLMKPGAFSKVRIVKIGGGVTDIVKEGEVGELIV
EAGDSCATGYLNAPALTAKRFIDGWYRTGDAAVWTADGSVEILGKIKDMIVSDGKPVFPAKVEEVLLEAEG
VKDVVVIGIPNDDGTEDVTAVVVPEPGKTLSEEALLAFLKASDLPEYEIPKKVFIVDSLPKNELGRVLRKK
LKELILK

MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAV
LALLRLGAIPALIDPRLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFG
APVPVPARQPDDPAVIFYTGXXKGLPKAAVIPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFA
VLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKALAEAAADPNSTLDLSSLKHVTFAGATA
TDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTDVVAPGEAGELIV
ECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPG
VKEVAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGK
LKDLVLK

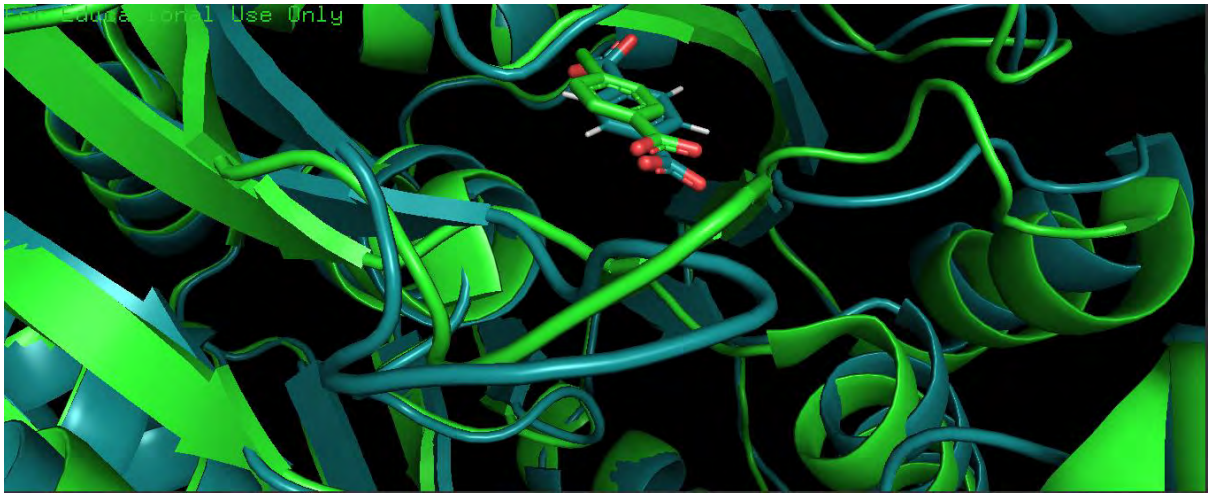
MKTVVEMVLEAAEKYPDQPALIDPDKGIELTHAEFLDAVDAVAARLAAMGLKPGERVAVISPNSLDTVIVI
LAILRLGAIPVLIDPRLPAEVIAELVKRDKCKAVVYDEGKEVKEAIEKSGSGAKIIPLEDLVRDGVPTVFG
PPVPLPERRPDEPYVIFYTSXXSGLPLGAVIPHRAAESRVLFMSTQVGLKPGTDNVILGLMPLYHVVGFFA
VLITALAFGGTLVVVEEFEPEKVLRLIQEEKVTSLFATPEQLAALAEAAADPNSDLDLSSLKHVTFAGATA
TEAVIARIRANLPGELVNIYGTTEAMNSLYKRNPENGRIMKPGAFSEVRIVKIGGGVTDVVKPGEVGELIV
QCGDSCATGYLNNPEATAKRFVDGWYRTGDAAVLTPDGEIEILGKIDDMIISNGKPIFPLRIEEVLLEAPG
VAEVKVIGLPNDNGEQDVTAVVVPKPGETLSEEALLAFLKESDLPEYEIPKKVVIVASLPRDHLGRILVDR
LKEMVAS

MKTVIDMVTEAAAKHPDHPALVDPAAGIALTHAEFMAAVRAVARRLAALGLKPGERVAVIAPNSLDTVLVI
LALLRLGAIPVLIDPRLPAELIAELVKHNRCRAAVISPGEEVEEAIRESGSGARVIRLEDLVRDGVPDAFG
PEIPEPEFRPDDPYVIFYTGXXEGLPLGAVIPHRAAESRVLFMSTQVGLKPGTDNRVLGLMPLYHVVGFFA
VLISGLAFGGTVVVVREFDAERVLQLVQDEKVTSLFATPEQLAALAAAASDPNSSLDLSSLKYVTFAGATA
TDAQIASIKKNLPGTLVNIYGTTEAMNSLYKVNPTNGRTMKPGAFSEVRIVKVGGGVDDVVEPGEDGELIV
KAGDSCATGYLNRPEATAERFVDGWYRTGDIARWLDDGTIEILGRIKDMIISNGKRIYPQKVEDVILTFPG
VADAVVIGIPDDNGTEEVTAVVVPKPGETLSEDAILAFCKKSDLPEYEIPKKVFIVDSLPRNHLKRIKREE
LKELISS

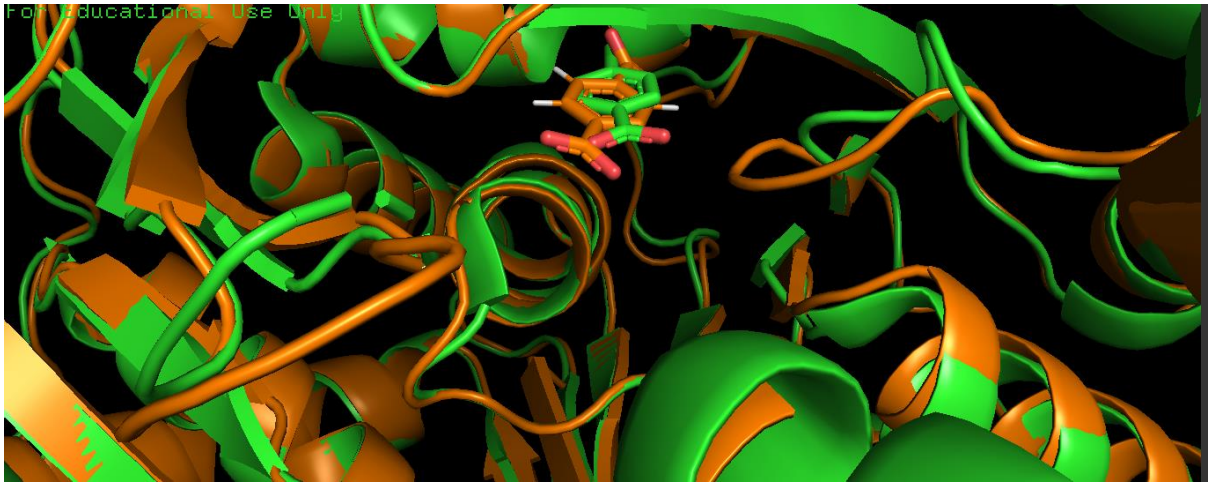
- Question now is how to proceed. I want to do docking, but for that I need a 3D structure. Also the residue number for the missing residue is 164 residues.
- Edu has the 3D structures!
- Will do docking on each of them.
- IMPORTANT: The labeling above might not be correct. Unsure how Edu structure his files and if the 3D structure came in the same order. Will rename.
- I have a bit of a problem, the generated sequences does not have correct coordinates so the extractions box needs to be modified.
- Figured out I can just align all of them with the crystal structure to make the box fit.

- I have performed the dockings with varied results. Will download the best ones and compare with crystal structure. Thinking cofolding might be a good idea to see if it provides any other results.
- Summarized results.
    - Seq A:
        - No good results. Not in correct pocket, low affinity and varying strain
    - Seq B:
        - Two dockings in correct position, both with good binding affinity but also relatively high strain compared to previous dockings.
    - Seq C:
        - Two dockings in correct position, both have good binding affinity and but also relatively high strain.
    - Seq D:
        - No good dockings. See A.
    - Seq E:
        - No good dockings. See A.
- After evaluating in pymol: No perfect result. Would say that the dockings on structure C matches better with the docked crystal structure.

Final seq C pose 2 (dark green):



Final seq B pose 2 (orange):



- Conclusion: I will proceed with the two good structures. We are limited anyways with how many we can order. Will still do cofolding on all of the structures.

- When doing the cofolding you need the sequence. I do not trust the ones above. They do not match with the cif files Edu uploaded on the drive. Super weird. Will have to double check which ones are the correct ones, the ones shared above (second iteration of A, B, C, D, E) or the ones uploaded to drive. Almost want to check this before proceeding, because otherwise it will be a lot of work for nothing (if I dock and cofold the wrong structures).
- We found a fault in the MPNN code and the above sequences cannot be guaranteed to be the correct ones. The docked structures are wrong, so will do it again.

Code

None relevant

Results Summary

- Initial summarized results.
    - Seq A:
        - No good results. Not in correct pocket, low affinity and varying strain
    - Seq B:
        - Two dockings in correct position, both with good binding affinity but also relatively high strain compared to previous dockings.
    - Seq C:
        - Two dockings in correct position, both have good binding affinity and but also relatively high strain.
    - Seq D:
        - No good dockings. See A.
    - Seq E:
        - No good dockings. See A.
- We found a fault in the code and the above sequences cannot be guaranteed to be the correct ones. The docked structures are wrong, so will do it again

Interpretation

- None, experiment not fruitful
    - Sequences do not match the cif files from Edu

Next Steps

- Fix the code and redo experiment
- Try cofolding
- Compares docked and cofolded structures with crystal structure
- Perform mutations and repea

2025-07-22 Co-folding 1T5D MPNN structures, the right sequences this time

Contributors: Sixten

Objective

Cofold in Rowan to see if 4-coumarate binds in a promising way to 3TSY.

- Today's idea:
    - Align the cofolded structure of 3tsy and compare it to the crystal structure.

Background

(see entry 2025-07-18, 21) Since we have ordered 3TSY which is apparently a fusion protein (whoops) we need to do some checks that the substrate binds in the correct way to 3TSY.

Methods and parameters

Component	*Description*
Software Used	pyMOL, Rowan
Input Structure	3TSY (sequences given below)
Key Parameters	Substrate position and orientation in the protein
Script Location	None

Used Rowan GUI to perform co-foldings with 4-coumarate.

- This is cofolded sequence:
  MAPQEQAVSQVMEKQSNNNNSDVIFRSKLPDIYIPNHLSLHDYIFQNISEFATKPCLI
  NGPTGHVYTYSDVHVISRQIAANFHKLGVNQNDVVMLLLPNCPEFVLSFLAASFRG
  ATATAANPFFTPAEIAKQAKASNTKLIITEARYVDKIKPLQNDDGVVIVCIDDNESVIPE
  GCLRFTELTQSTTEASEVIDSVEISPDDVVALPYSSGTTGLPKGVMLTHKGLVTSVA
  QQVDGENPNLYFHSDDVILCVLPMFHIYALNSIMLCGLRVGAAILIMPKFEINLLLELIQ
  RCKVTVAPMVPPIVLAIAKSSETEKYDLSSIRVVKSGAAPLGKELEDAVNAKFPNAKL
  GQGYGMTEAGPVLAMSLGFAKEPFPVKSGACGTVVRNAEMKIVDPDTGDSLSRNQ
  PGEICIRGHQIMKGYLNNPAATAETIDKDGWLHTGDIGLIDDDDELFIVDRLKELIKYK
  GFQVAPAELEALLIGHPDITDVAVVAMKEEAGEVPVAFVVKSKDSELSEDDVKQFVS
  KQVVFYKRINKVFFTESIPKAPSGKILRKDLRAKLANGL

- Sequence generated by ChatGPT from pdb. Does not seem correct so lets not do this in future.

  DVIFRSKLPDIYIPNHLSLHDYIFQNISEFATKPCLINGPTGHVYTYSDVHVISRQIAAN
  FHKLGVNQNDVVMLLLPNCPEFVLSFLAASFRGATATAANPFFTPAEIAKQAKASNT
  KLIITEARYVDKIKPLQNDDGVVIVCIDDNESVPIPEGCLRFTELTQSTTEASEVIDSVE
  SPDDVVALPYSSGTTGLPKGVMLTHKGLVTSVAQQVDGENPNLYFHSDDVILCVLP
  MFHIYALNSIMLCGLRVGAAILIMPKFEINLLLELIQRCKVTVAPMVPPIVLAIAKSSETE
  KYDLSSIRVVKSGAAPLGKELEDAVNAKFPNAKLGQGYGMTEAGPVLAMSLGFAKE
  PFPVKSGACGTVVRNAEMKIVDPDTGDSLSRNQPGEICIRGHQIMKGYLNNPAATA
  ETIDKDGWLHTGDIGLIDDDDELFIVDRLKELIKYKGFQVAPAELEALLIGHPDITDVA
  VVAMKEEAGEVPVAFVVKSKDSELSEDDVKQFVSKQVVFYKRINKVFFTESIPKAPS
  GKILRKDLRAKLANGL

- Sequnce generated from a pdb2fasta website. Seems to be correct.

  DVIFRSKLPDIYIPNHLSLHDYIFQNISEFATKPCLINGPTGHVYTYSDVHVISRQIAAN
  FHKLGVNQNDVVMLLLPNCPEFVLSFLAASFRGATATAANPFFTPAEIAKQAKASNT
  KLIITEARYVDKIKPLQNDDGVVIVCIDDNESVPIPEGCLRFTELTQSTTEASEVIDSVE
  ISPDDVVALPYSSGTTGLPKGVMLTHKGLVTSVAQQVDGENPNLYFHSDDVILCVLP
  MFHIYALNSIMLCGLRVGAAILIMPKFEINLLLELIQRCKVTVAPMVPPIVLAIAKSSETE
  KYDLSSIRVVKSGAAPLGKELEDAVNAKFPNAKLGQGYGMTEAGPVLAMSLGFAKE
  PFPVKSGACGTVVRNAEMKIVDPDTGDSLSRNQPGEICIRGHQIMKGYLNNPAATA
  ETIDKDGWLHTGDIGLIDDDDELFIVDRLKE

- Sequence from pymol. Probably most trustworthy

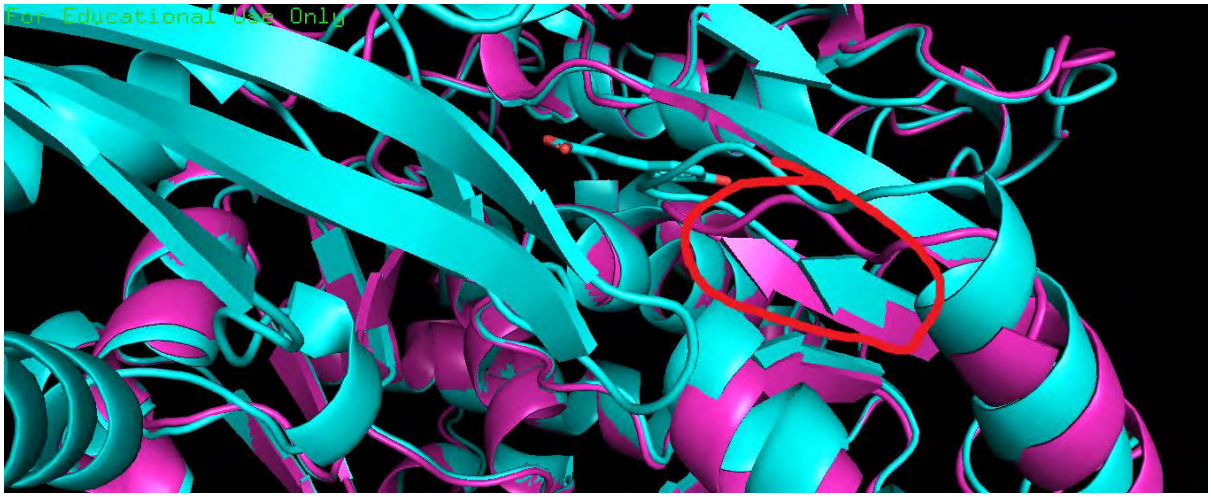
  DVIFRSKLPDIYIPNHLSLHDYIFQNISEFATKPCLINGPTGHVYTYSDVHVISRQIAAN
  FHKLGVNQNDVVMLLLPNCPEFVLSFLAASFRGATATAANPFFTPAEIAKQAKASNT
  KLIITEARYVDKIKPLQNDDGVVIVCIDDNESVPIPEGCLRFTELTQSTTEASEVIDSVE
  ISPDDVVALPYSSGTTGLPKGVMLTHKGLVTSVAQQVDGENPNLYFHSDDVILCVLP
  MFHIYALNSIMLCGLRVGAAILIMPKFEINLLLELIQRCKVTVAPMVPPIVLAIAKSSETE
  KYDLSSIRVVKSGAAPLGKELEDAVNAKFPNAKLGQGYGMTEAGPVLAMSLGFAKE
  PFPVKSGACGTVVRNAEMKIVDPDTGDSLSRNQPGEICIRGHQIMKGYLNNPAATA
  ETIDKDGWLHTGDIGLIDDDDELFIVDRLKE

Code

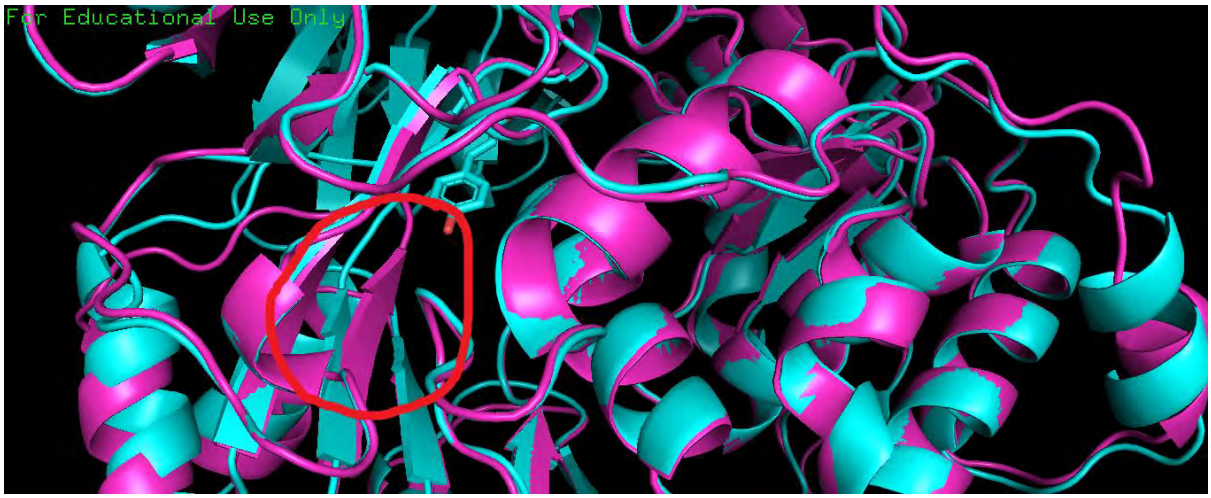
None

Results Summary

Aligned structures (cofolded structure is cyan and crystal structure is purple):

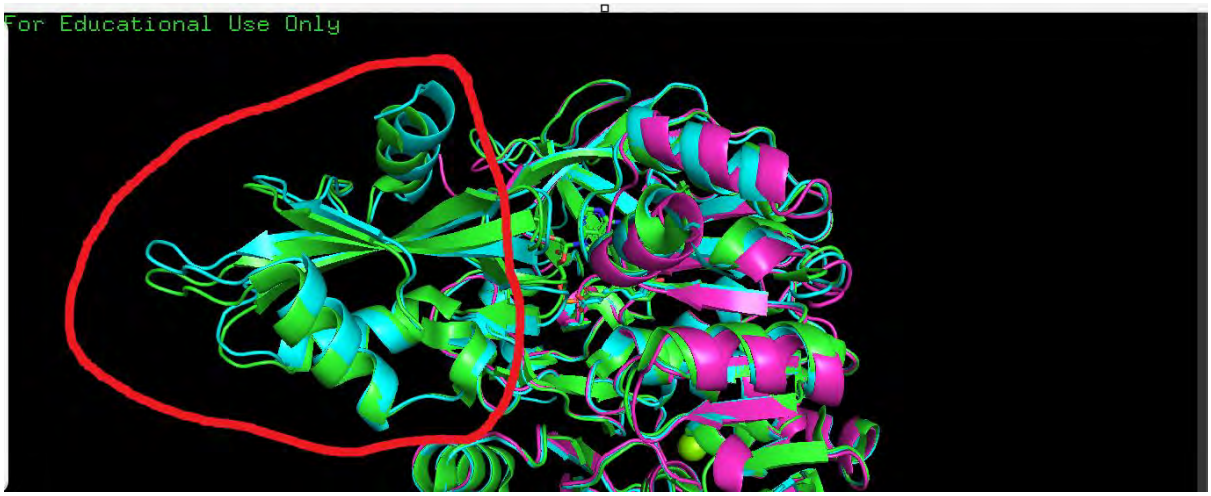


Different view:



- The biggest difference between structures close to the active site are highlighted. Could maybe affect docking.
- Wait honestly this might be super dumb, but the crystal structure of 3tsy actually misses a big chunk of the protein, do not know how this affect docking

5bst (green), cofolded 3tsy (cyan), and crystal 3tsy (purple)



Interpretation/Discussion

- The crystal structure of 3tsy actually misses a big chunk of the protein, do not know how this affect docking

Next Steps

- Investigate reasons why 3tsy cofolded structure is different from crystal structure?

2025-07-22 Evaluation of 1T5D proteinMPNN sequences by docking and co-folding

Contributors: Sixten

Objective

Investigate all of the generated sequences from proteinMPNN, contact Alma about 3TSY and also organize so we know exactly what we want the wet lab team to order.

Background

- What we want to order:
    - 1 sequence containing the most promising mutation of 1t5d (only need primer)
    - 1 sequence containing the most promising mutation of 5bst (only need primer)
    - 2 sequences generated from protein MPNN containing the most promising mutation for 1t5d
    - 2 sequences generated from protein MPNN containing the most promising mutation for 5bst
    - The insert plus plasmid for 5bst.

Methods and parameters

Component	*Description*
Software Used	FoldX
Input Structure	proteinMPNN sequences made from 1T5D, given below. Ligands: TPA and 4-chlorobenzoate.
Key Parameters	Docking score, ligand strain, alignment of ligand position and orientation with crystal structure, pLDDT and ligand affinity (co-folding)
Script Location	

Sequences (A-E)

MKTVVDMIREAARHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAV
IPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTD

VVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVKE
VAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

MKTVVDMIREAARHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAVI
PHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVT
DVVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVK
EVAVIGLPNDDGSEDVTAVVVPEPGKTLSEEMAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAV
IPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTD
VVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVKE
VAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAV
IPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTD
VVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVKE
VAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAV
IPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTD
VVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVKE
VAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

2025-07-23

- Now investigating the sequences generated by proteinMPNN from 1T5D. Hopefully it is the correct ones this time.
- Results from docking:
    - Seq A: 2 potentially ok dockings. They were docked in correct positions but binding affinity is low.
    - Seq B: Potentially three good dockings
    - Seq C: Potentially two good dockings
    - Seq D: Potentially two good dockings
    - Seq E: Potentially two good dockings
- Investigated all mutations in pyMOL.
    - Seq A: 2 ok dockings. They were docked in correct positions but binding affinity is low.
    - Seq B: Three good dockings. Very high binding affinities but relatively high strain
    - Seq C: Two good dockings. Very high binding affinities but relatively high strain
    - Seq D: Two good dockings. Very high binding affinity and relatively high strain
    - Seq E: The ligand does not seem to match position with crystal structure well enough.
- Another comment is that they are all a tiny bit off the crystal structure. C or D looks the best however.
- Will proceed to cofolding. Need fasta for all sequences. Will do this in pymol. Important to note here is that there are missing residues (two) in the sequence and

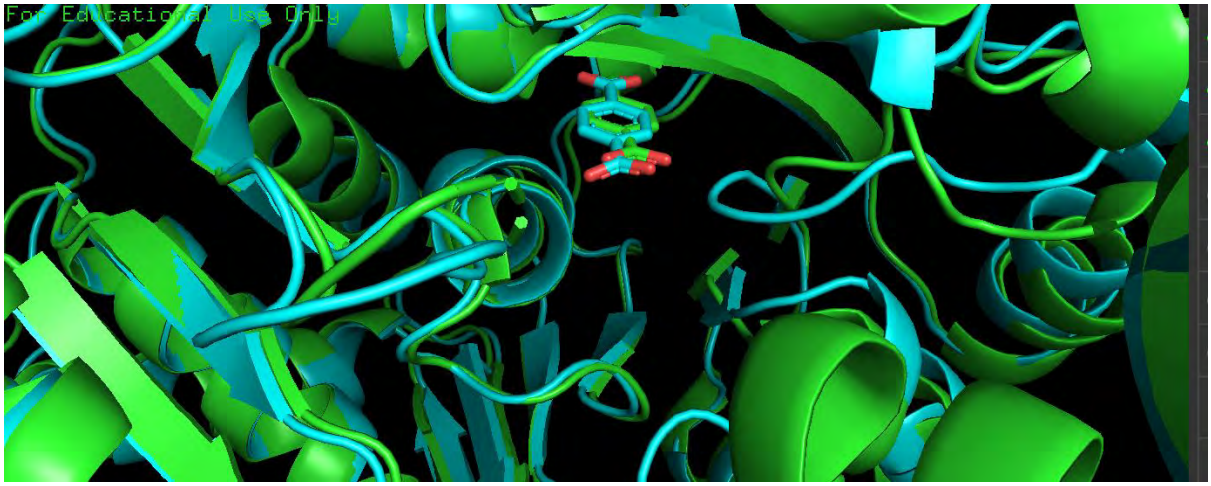
that in pyMOL they are denoted as ??. However, by changing these to XX it works. Will do this.

- Just as a note: The missing residues seem to be residue 163 and 164.

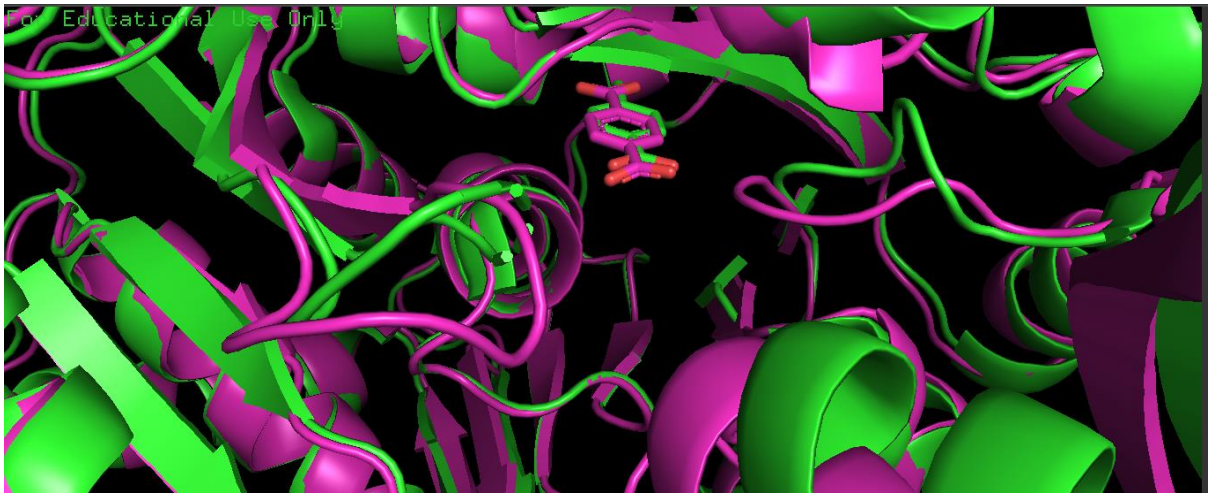
MKTVVDMIREAAERHPDAPALVDPKRGIRLTHAEFLAAVDAVAQRLASMGLQPGERVGVVASNSLDTVLAVLALLRLGAIPALIDP
RLPAEVIAELVKRDGLKAVVYSEGAEVAKAIEESGSGAKLIPLSSLVRDGVPRAFGAPVPVPARQPDDPAVIFYTGXXKGLPKAAV
IPHQAAESRVLFMSTQVGLKPGTDNVVLGLMPLYHVVGFFAVLVQALAFGGTLVVVEEFDAEDALRLVETEKVTSLFATPEQLKA
LAEAAADPNSTLDLSSLKHVTFAGATATDEQIAAIKKNLPGRLVNIYGTTEAMNSLYKVDPTNGTTMKPGAFSKVRIVKVGGGVTD
VVAPGEAGELIVECGESCATGYLNDPEATAKRFVDGWYRTGDMAVRTASGEIEILGKIERMIISKGKPIYPEKVEKVLLKAPGVKE
VAVIGLPNDDGSEDVTAVVVPEPGKTLSEEAILAYLAASSLPEYEKPKKVFILDSLPKDELNRIQYGKLKDLVLK

- Also, checked if the mutations from the script that generates mutations are correct and they seem good.
- It appears I will run out of credits in Rowan.
- The cofolding for B and C is not working correctly (ran for approx 35 minutes without result). Will do some investigation of the sequences.
- Apparently it works now. Do not know why. The sequence for B was actually wrong in Rowan, so i guess this shows how scary it is to just copy and paste everything manually (one single error can f up everything and the chance is pretty high). This is a reason why automated systems are so much better.
- All cofolding results align well with crystal structure ligand except C. A, B and E aligned the best. Would like to align the cofolded structures with the pdbs.

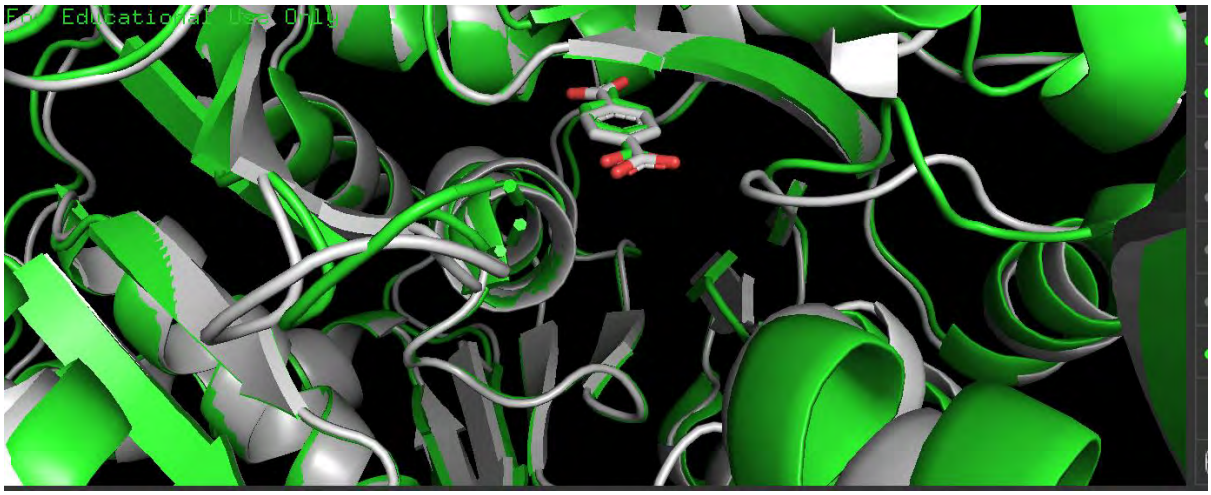
Cofold A (cyan)



Cofold B (purple):

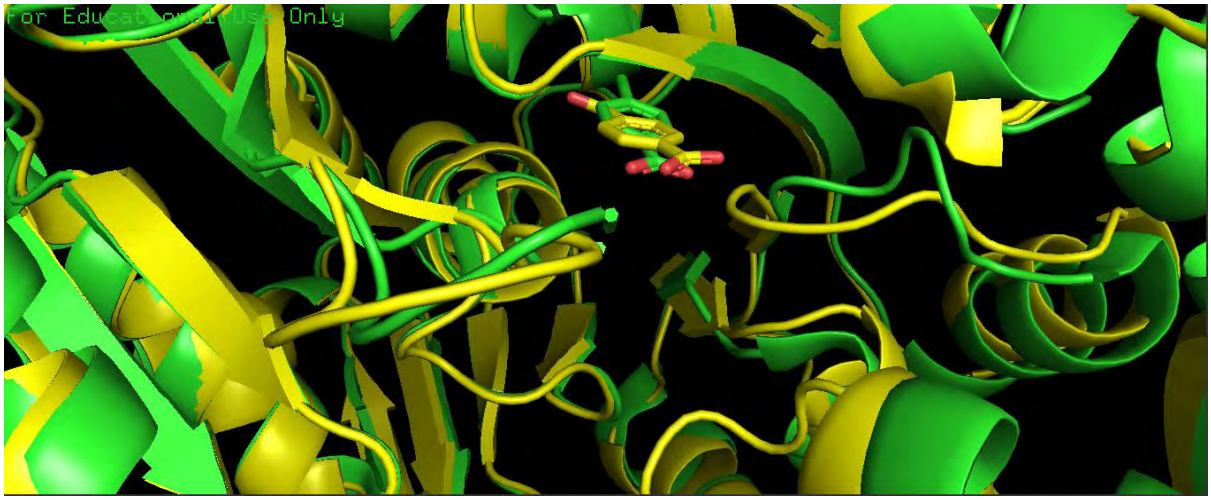


Cofold E (white):

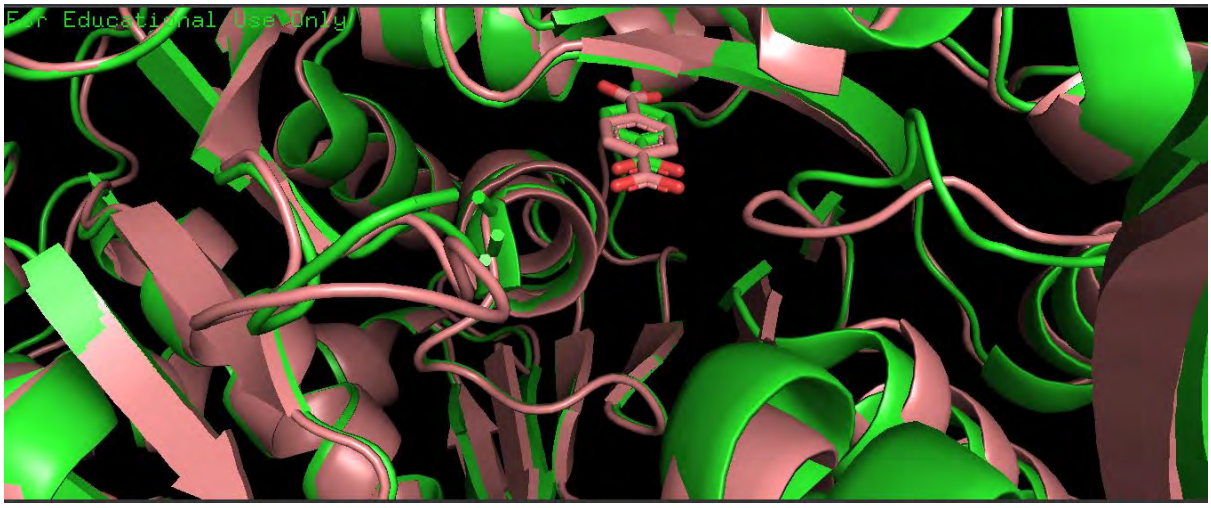


- And to give you the not so good foldings

Cofold C (yellow):

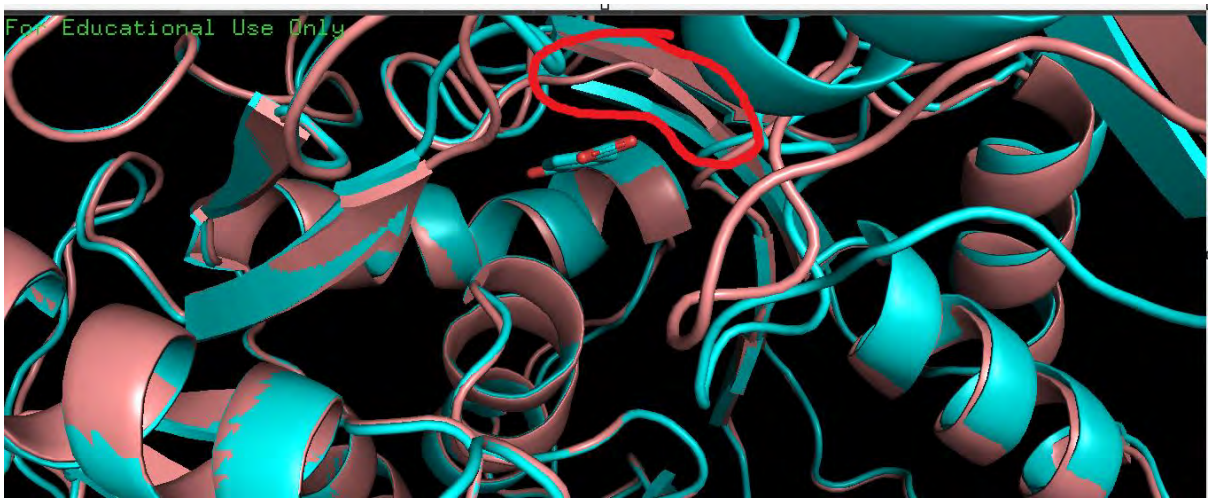


Cofold D (I dono what color this is but crystal is green at least). This one looks alright but not quite as good as the other ones.



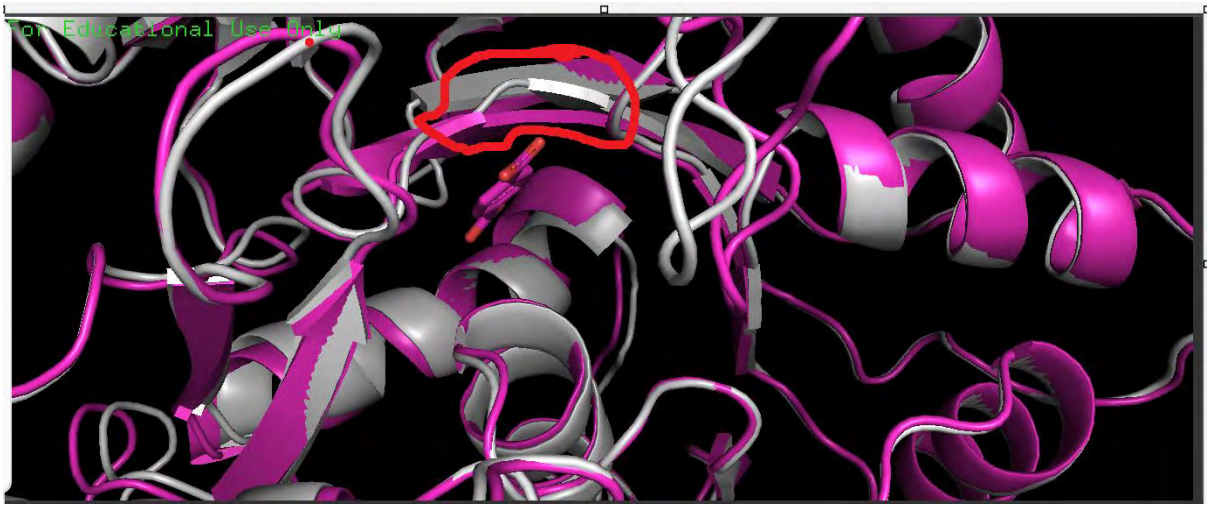
- Comparing cofolded structures from Rowan to the pdb structures generated by chai.
    - A:s active site looks super similar
    - B:s active site is a bit different (highlighted red)

Cofolded structure is cyan and chai structure is the question mark colour.



    - C:s active site is different in the same way as B.

Cofolded structure is purple and chai structure is white:



- A general thought is that that beta sheet has been varying quite a bit throughout the project so maybe a point of interest.
- Now I want to try the point mutation on all sequences.

Code

Script for docking using Rowan API (see previous entries). Co-folding performed with GUI, no scripts used.

Results Summary

To organize the result and compare them to previous dockings I will put everything in a table:

Protein	Docking score first docking (best result)	Docking score from mutated docking (best result)	Strain from first docking (best result) (kcal/mol)	Strain from mutated docking (best result) (kcal/mol)	pLDDT and affinity probability from cofolding	pLDDT and affinity probability from cofolding mutated structure
A	-2,8 (pose 1)	-3,6 (pose 1)	2,482	5,521	0,934 and 0,425	0,936 and 0,493
B	-3,1 (pose 3)	-4,0 (pose 1)	0,441	1,149	0,923 and 0,481	0,937 and 0,539
C	-3,7 (pose 1)	-4,0 (pose 1)	1,468	2,272	0,938 and 0,371	0,937 and 0,483
D	-3,541 (pose 2)	-3,9 (pose 3)	1,011	1,849	0,943 and 0,397	0,942 and 0,520
E	-3,0 (pose	-3,2 (pose 2)	1,313	3,555	0,943 and	0,942 and

Protein	Docking score first docking (best result)	Docking score from mutated docking (best result)	Strain from first docking (best result) (kcal/mol)	Strain from mutated docking (best result) (kcal/mol)	pLDDT and affinity probability from cofolding	pLDDT and affinity probability from cofolding mutated structure
	1)				0,450	0,515

- Summary: For the docking it can be concluded that the docking score consistently increases and that the strain consistently increases. This leads us to the conclusion that the ligand binds stronger in the pocket but has to strain itself to fit appropriately. When it comes to the cofolding it can be concluded that the pLDDT stays approximately the same and that the affinity probability consistently increases.

Interpretation

- The dockings and cofolds were evaluated in pymol and it can be concluded that all of the docked ligands are in the same plane as the crystal structure ligand but rotates a bit clockwise. The cofolds align very good with the ligand of the crystal structure.
- Final conclusion is that B and D seems to be the best ones overall when taking both docking and cofolding into account.
- Good results! Docks in the same position and only slightly lower binding.

Next Steps

- Need to make a plan for Coumarate: Since docking with the 3TSY structure does not work properly we will proceed with the 5BST one.
- Optimize workflow with automation:
    - It becomes really apparent that optimizing the workflow by automating both mutations and docking and cofolding would be incredible. In theory we could do it but since we need to order the sequences due to time constraints we do not have a real reason to automize. We could however automize it to give other iGEM teams code and workflow that will aid in their work.

2025-07-24 Comparing docked structure of 5BST with native substrate vs TPA

Contributors: Sixten

Objective

Investigate the binding site of 5BST and how it binds substrates. Compare the crystal structure of 5BST with structures of 5BST docked with 4-chlorobenzoate and TPA.

Background

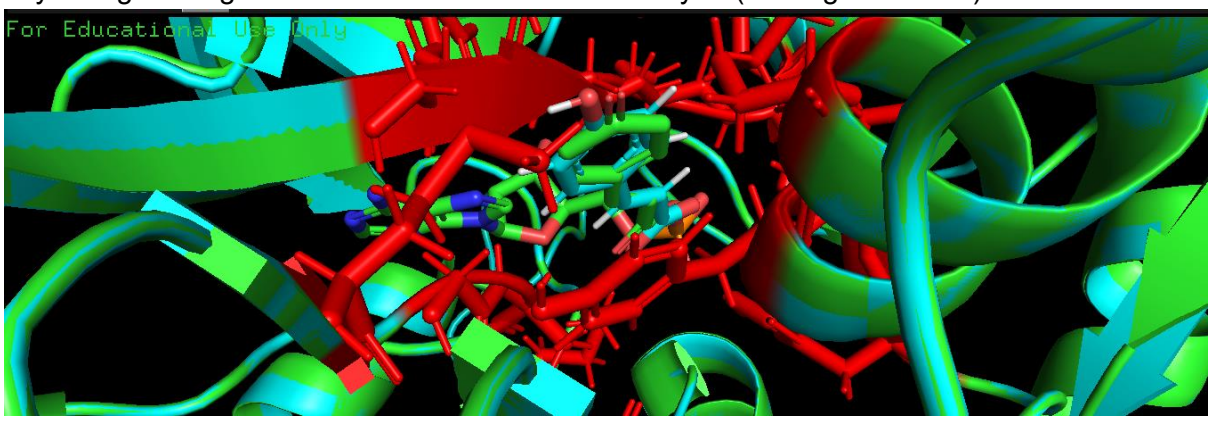
Before testing mutations, we want to make sure that we are able to dock both the native substrate and TPA in the correct site in 5BST. Visualisation will help see the differences on and understand how the site interacts with the different ligands.

Methods and parameters

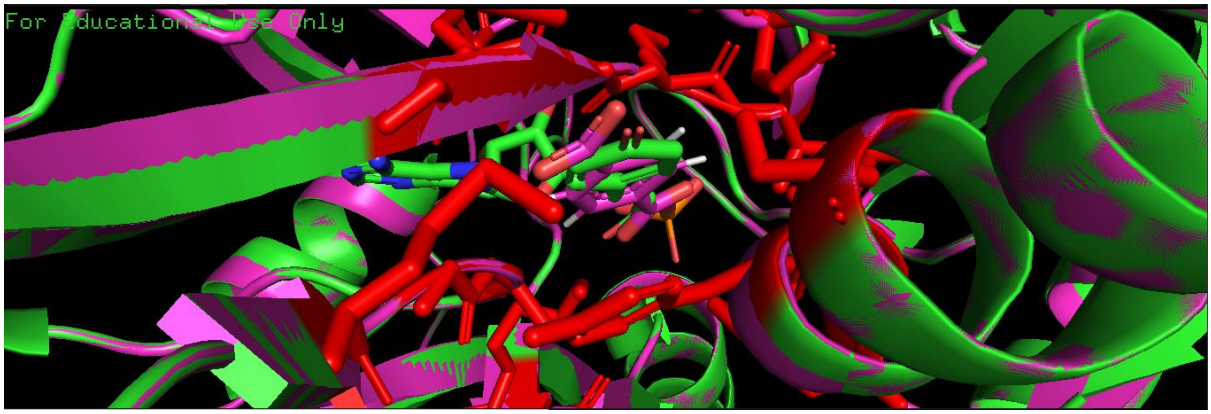
Component	*Description*
Software Used	Rowan pyMOL
Input Structure	5BST Ligand: TPA, Coumarate
Key Parameters	Ligand position and orientation, docking score, ligand strain
Script Location	Binding site selector

- Investigated the coumarate a bit yesterday. These was the docking:

Crystal ligand is green and docked 4-coumarate is cyan (binding site is red):



Crystal ligand is green and docked TPA is purple (binding site is red):



Code

For binding site selection script see entry 2025/07/13

Results Summary

Success for both substrates. Some differences in orientation.

Interpretation

Looks promising for further investigations.

Next Steps

- How to proceed:
    - Test mutations (see point mutations list)
    - Dock to see if any good candidates (see point mutations list for notes and results)
- I remembered I had a script for mutations. Let's do that one actually.

2025-07-25 Evaluating ddG of 5BST and 1T5D mutants using FoldX

Contributors: Hilya

Objective

Calculate ddG of 5BST mutants to evaluate effect of mutation to stability. Will evaluate 6 residues in the active site with all possible 19 mutations.

Background

(See entry 2025-07-22) One of the effects of mutations on an enzyme is the impact to the stability of the enzyme. Stability can be assessed by calculating the free energy difference (ΔG) of unfolding. A mutation can stabilising or destabilising depending on the difference of the ΔG unfolding of the mutant and the wild type.

Reference: Bromberg, Y., & Rost, B. (2009). Correlating protein function and stability through the analysis of single amino acid substitutions. BMC Bioinformatics, 10(S8). https://doi.org/10.1186/1471-2105-10-s8-s8

Methods and parameters

Component	*Description*
Software Used	FoldX
Input Structure	5BST, repaired with FoldX Mutations generated with script
Key Parameters	Total Energy
Script Location	FoldX

- Created a script to generate mutants in an individual_list.txt file (but still manual input). **Might be better if we can specify the output file name.**
- Generated individual_list.txt for coumarate mutations (114 mutants)
- Repaired 5bst.pdb with FoldX
- Performed calculations for 114 mutations for 5bst with number of runs set to 5
    - but got error “Sorry yo specified an option_FOLDXDEFINED for the order of mutations that does not exist, should be _USERDEFINED or _FOLDXDEFINED”
- Process completed and collected everything in a table.
- Repeated the process for 1t5d with 95 mutants.

- Uploaded all results.
- Asked DeepSeek to summarise results from the csv files.

Code

Script for mutations 2 (added 2025-07-25)

Script for making mutations in a FoldX individual list file. Takes manual input (not automated from distance)

```
#!/usr/bin/env python3 #(don't forget to change to your venv)
import sys

# Three-letter to one-letter amino acid conversion
AA_CONVERSION = {
    "Ala": "A", "Arg": "R", "Asn": "N", "Asp": "D", "Cys": "C",
    "Gln": "Q", "Glu": "E", "Gly": "G", "His": "H", "Ile": "I",
    "Leu": "L", "Lys": "K", "Met": "M", "Phe": "F", "Pro": "P",
    "Ser": "S", "Thr": "T", "Trp": "W", "Tyr": "Y", "Val": "V"
}

def main():
    if len(sys.argv) < 2:
        print("Usage: ./generate_mutants.py <ThreeLetterAA><resnum><chain> [residues...]")
        print("Example: ./generate_mutants.py Met306A Gly332A Met344A Gln213A Ser243A
Val341A")
        sys.exit(1)

    residues = []
    for arg in sys.argv[1:]:
        # Extract three-letter code (first 3 characters)
        aa_three = arg[:3]
        # The rest is <resnum><chain>
        num_chain = arg[3:]

        # Separate digits from chain letter
        res_num = ''.join(filter(str.isdigit, num_chain))
        chain = num_chain[len(res_num):]

        if not res_num:
            print(f"Error: Missing residue number in '{arg}'")
            sys.exit(1)
        if not chain:
            print(f"Error: Missing chain identifier in '{arg}'")
            sys.exit(1)
```

```
        # Convert to one-letter code
        try:
            aa_one = AA_CONVERSION[aa_three]
        except KeyError:
            print(f"Error: Unknown amino acid '{aa_three}' in '{arg}'")
            sys.exit(1)

        residues.append((aa_one, chain, res_num))

    # All possible amino acids (one-letter codes)
    all_aas = "ACDEFGHIKLMNPQRSTVWY"

    # Generate mutations
    mutations = []
    for wt, chain, num in residues:
        for mut in all_aas:
            if mut != wt:
                mutations.append(f"{wt}{chain}{num}{mut};")

    # Write to file
    with open("individual_list.txt", "w") as f:
        f.write("\n".join(mutations))

    print(f"Generated individual_list.txt with {len(mutations)} mutations")

if __name__ == "__main__":
    main()
```

Script for reading FoldX Dif Outputs (added 2025-07-25)

```
#!/usr/bin/env python3 #change to your venv!
import sys
import re
import csv
import os

# Amino acid conversion dictionaries
ONE_TO_THREE = {
    'A': 'Ala', 'R': 'Arg', 'N': 'Asn', 'D': 'Asp', 'C': 'Cys',
    'Q': 'Gln', 'E': 'Glu', 'G': 'Gly', 'H': 'His', 'I': 'Ile',
    'L': 'Leu', 'K': 'Lys', 'M': 'Met', 'F': 'Phe', 'P': 'Pro',
    'S': 'Ser', 'T': 'Thr', 'W': 'Trp', 'Y': 'Tyr', 'V': 'Val'
}

def parse_mutation(mut_str):
    """Convert FoldX mutation string to descriptive format"""
    clean_str = mut_str.strip().rstrip(';')
    wt = ONE_TO_THREE.get(clean_str[0], clean_str[0])
```

```
    chain = clean_str[1]
    residue_num = ''.join(filter(str.isdigit, clean_str[2:]))
    mut_aa = ONE_TO_THREE.get(clean_str[-1], clean_str[-1])
    return f"{wt}{residue_num} to {mut_aa}, Chain {chain}"

def main():
    if len(sys.argv) != 3:
        print("Usage: ./parse_foldx.py <mutations_file> <foldx_output>")
        print("Example: ./parse_foldx.py individual_list.txt FoldX_output.txt")
        sys.exit(1)

    # Read mutation list
    with open(sys.argv[1], 'r') as f:
        mutations = [line.strip() for line in f if line.strip()]

    # Parse FoldX output
    results = []
    data_started = False

    with open(sys.argv[2], 'r') as f:
        for line in f:
            # Skip version and header lines
            if not data_started:
                if line.startswith("Pdb") and "total energy" in line:
                    data_started = True
                continue

            # Skip empty lines
            if not line.strip():
                continue

            # Split line by tabs
            parts = line.split('\t')
            if len(parts) < 2:
                continue

            # First element is filename
            filename = parts[0].strip()
            # Second element is total energy
            energy = parts[1].strip()

            # Extract mutant and run indices from filename
            match = re.search(r'_Repair_(\d+)_(\d+)\.pdb$', filename)
            if not match:
                print(f"Warning: Could not parse filename '{filename}'")
                continue

            mutant_idx = int(match.group(1))
```

```
            run_idx = int(match.group(2)) + 1  # Convert to 1-based

            # Get mutation description
            if 1 <= mutant_idx <= len(mutations):
                mut_desc = parse_mutation(mutations[mutant_idx-1])
                results.append((mut_desc, run_idx, energy))

    if not results:
        print("Error: No valid mutations found in the output file")
        print("Check if your mutation file matches the FoldX output")
        sys.exit(1)

    # Create output filename
    input_basename = os.path.splitext(os.path.basename(sys.argv[2]))[0]
    output_file = f"{input_basename}_energies.csv"

    # Write to CSV
    with open(output_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(["Mutation", "Run", "Total_Energy"])
        for mut_desc, run_idx, energy in results:
            writer.writerow([mut_desc, run_idx, energy])

    print(f"Successfully created {output_file} with {len(results)} energy entries")
    print(f"First mutation: {results[0][0]}, Run {results[0][1]}, Energy {results[0][2]}")

if __name__ == "__main__":
    main()
```

Results Summary

1T5D

https://drive.google.com/file/d/1dYCZDKHS-mBWt7AfDETPMdf2jv4ToNNm/view?usp=sharing

Top 5 Most Stabilizing Mutants (Negative ΔG)

Rank	Mutation	Average Energy (kcal/mol)	Notes
1	Gly305 to Ala, X	-0.42	Stabilizing (only Run 1 is negative)
2	Ile303 to His, X	-0.019	Slight stabilization
3	Ile303 to Lys, X	-0.019	Slight stabilization

4	Ile303 to Leu, X	-0.019	Slight stabilization
5	Ile303 to Met, X	-0.019	Slight stabilization

Note: The stabilization for Ile303 to His/Lys/Leu/Met is very slight. Asn311 to His and others have a value of +0.014, which is neutral/very slightly destabilizing.

Top 5 Most Destabilizing Mutants (Positive ΔG)

Rank	Mutation	Average Energy (kcal/mol)	Notes
1	Phe184 to Ala, X	3.08	Highly destabilizing (only Run 1 non-zero)
2	Ile303 to Ala, X	3.08	Highly destabilizing (only Run 1 non-zero)
3	Met310 to Ala, X	2.19	Destabilizing (Runs 1 & 2 non-zero)
4	Phe184 to Gly, X	1.24	Consistently destabilizing across all 5 runs
5	Asn311 to Gly, X	1.09	Consistently destabilizing across all 5 runs

5BST
https://drive.google.com/file/d/1G-ovFhcmCdfNLQ9HRpowdYPI45VmyigX/view?usp=sharing

Top 5 Most Stabilizing Mutants (Negative ΔG)

Rank	Mutation	Average Energy (kcal/mol)	
1	Met306 to His, A	-1.58	Strong, consistent stabilization
2	Met344 to His, A	-1.23	Strong, consistent stabilization
3	Ser243 to Ala, A	-0.011	Slight stabilization (only Run 1 is negative)

Note: Only three mutations in the 5BST dataset showed a net stabilizing effect. The other mutations to His, Ile, Lys, etc., all have an average energy of +0.00059, which is effectively neutral.

Top 5 Most Destabilizing Mutants (Positive ΔG)

Rank	Mutation	Average Energy (kcal/mol)	Notes

1	Met344 to Ala, A	3.17	Highly destabilizing (only Run 1 non-zero)
2	Met306 to Ala, A	2.12	Highly destabilizing (only Run 1 non-zero)
3	Ser243 to Gly, A	1.52	Consistently destabilizing across all 10 runs
4	Met306 to Gly, A	1.58	Destabilizing (only Run 1 non-zero)
5	Met344 to Gly, A	1.24	Destabilizing (only Run 1 non-zero)

Summary:

- 5BST has more pronounced and energetically significant stabilizing mutations (e.g., Met to His).
- 1T5D has more mutations that are consistently destabilizing across all runs (e.g., Phe184 to Gly).
- For both structures, substitutions to Alanine (to Ala) are by far the most destabilizing changes, as they remove most of the side chain.
- Substitutions to Glycine (to Gly) are also consistently destabilizing, as they introduce excessive backbone flexibility.
- Many mutations only have a non-zero value in the first run. This is a common artifact in computational mutagenesis where subsequent runs might not find a lower energy conformation, or the calculation may have been set up to only fully minimize the first run.

Interpretation/Discussion

Found some stabilising and destabilising mutations. Many of the energies are zero. Could it mean that either the change is too small (mutation neither stabilising or destabilising) or that the process got stuck into a local minima?

Some discussion points from DeepSeek:

- Alanine Substitutions are Highly Destabilizing: The top destabilizing mutations are consistently substitutions to Alanine (to Ala). This is a common result in computational mutagenesis, as Ala removes all side-chain atoms beyond the beta carbon, often abolishing important interactions.
- Glycine Substitutions are Also Destabilizing: Substitutions to Glycine (to Gly) are the next most common destabilizing change. Glycine introduces extreme backbone flexibility, which can disrupt the precise packing of the protein structure.
-
- Stabilizing Mutations are Rare: Only a handful of mutations show a stabilizing effect. The most significant are Histidine substitutions (to His) at methionine residues in the 5BST structure.
- Neutral Mutations: The vast majority of mutations (e.g., to Cys, Asp, Glu, Phe, Ile, Leu, etc.) result in a calculated energy change of 0, suggesting they are well-tolerated or that the simulation did not converge on a non-zero value for those variants.

- Data Consistency: For many mutations, only one run produced a non-zero value while all others were zero. This could indicate that those runs failed to properly minimize or that the mutation truly has no effect in most conformational samples. The averages for these are based on the single non-zero value.

Next Steps

- Figure out the meaning of the error “Sorry yo specified an option_FOLDXDEFINED for the order of mutations that does not exist, should be _USERDEFINED or _FOLDXDEFINED”
- We want to try FoldX for coumarate mutations.
- Edu has a python script to identify active site residues.
- Will develop script to generate the list of mutations we need (from examining an active site, through, say, a substrate-bound PDB file) instead of specifying it manually?
    - Edu will send notes later

2025-07-25 Docking of 1T5D mutations

Contributors: Sixten, Hilya

Objective

Evaluate all the mutations fo native 1T5D and pick the best top mutations.

Background

Using the Rowan API, it is possible to automate the docking. We want to test all possible mutations in the residues of interest in the native 1T5D protein.

Methods and parameters

Component	*Description*
Software Used	Rowan API, with python scripts in vscode
Input Structure	ProteinMPNN sequences A to E for 1T5D.
Key Parameters	Docking score, ligand strain
Script Location	Mutation generating script API script

- Created a folder with all the mutations for Ile 303 and Met 310 (38 x 5).
- Will do the dockings now in Rowan.

Code

Mutation script:

```
#Description:
#After loading your protein of interest in pymol and selected your
#residues of interest the script generates 19 mutatated pdbs for all residues
#and names them appropriately


from pymol import cmd
import os

# Settings
```

```
pdb_file = r"C:\Users\sixte\OneDrive\Desktop\Skola\Bioinformatics\Docking
project\ProteinMPNN 1t5d\Final sequnces PDB modified coordinates\E_1t5d.pdb"
selection = "selection_to_mutate"  # Your selection of residues
output_folder = "mutants_pmpnn_1t5d"
amino_acids = [
    "ALA", "ARG", "ASN", "ASP", "CYS",
    "GLN", "GLU", "GLY", "HIS", "ILE",
    "LEU", "LYS", "MET", "PHE", "PRO",
    "SER", "THR", "TRP", "TYR", "VAL"
]

# Make output folder
os.makedirs(output_folder, exist_ok=True)

# Extract residue info from selection
model = cmd.get_model(selection)
unique_residues = {(atom.chain, atom.resi, atom.resn) for atom in model.atom}

pdb_basename = os.path.basename(pdb_file).replace('.pdb', '')

for chain, resi, original_resn in unique_residues:
    resi_str = str(resi)
    for aa in amino_acids:
        if aa == original_resn:
            continue  # Skip original amino acid

        # Reload clean structure each time
        cmd.reinitialize()
        cmd.load(pdb_file, "protein")

        # Mutate
        cmd.wizard("mutagenesis")
        cmd.do(f"refresh_wizard")
        cmd.get_wizard().do_select(f"/protein//{chain}/{resi_str}/")
        cmd.get_wizard().set_mode(aa)
        cmd.get_wizard().apply()

        # Save
        out_name =
f"{pdb_basename}{chain}_{resi_str}_{original_resn}_to_{aa}.pdb"
        out_path = os.path.join(output_folder, out_name)
        cmd.save(out_path, "protein")
```

```
        cmd.set_wizard()  # Turn off wizard
```

API script:

```
#The script that talks with the Rowan API.
#Uploads and sanitizes the pdbs in the chosen folder and then proceeds to dock
#them with a chosen ligand


import rowan
import time
import json
import shutil
from pathlib import Path
from rowan import smiles_to_stjames, submit_docking_workflow

# Rowan API key
rowan.api_key = "INSERT API KEY HERE"

# --- Configuration ---
protein_folder =
Path(r"C:\Users\sixte\AppData\Local\Schrodinger\PyMOL2\mutants_pmpnn_1t5d")
ligand_smiles = "[O-]C(=O)c1ccc(C(=O)[O-])cc1"
workflow_base_name = "Docking_"
pocket_box = [[-19, 95, 18], [15, 17, 12]]  # Box coordinates


# Convert SMILES once
ligand_molecule = smiles_to_stjames(ligand_smiles)

# === STAGE 1: Submit all workflows ===
submitted_workflows = []

for pdb_file in protein_folder.glob("*.pdb"):
    protein_name = pdb_file.stem
    workflow_name = f"{workflow_base_name}{protein_name}"
    print(f"\n--- Submitting workflow for: {protein_name} ---")

    try:
        protein = rowan.upload_protein(protein_name, pdb_file)
        protein.sanitize()
```

```
        time.sleep(2)

        workflow = submit_docking_workflow(
            protein=protein.uuid,
            pocket=pocket_box,
            initial_molecule=ligand_molecule,
            do_csearch=True,
            do_optimization=True,
            name=workflow_name,
        )

        print(f"Submitted: {workflow.uuid}")
        submitted_workflows.append({
            "protein_name": protein_name,
            "pdb_path": str(pdb_file),
            "workflow_uuid": workflow.uuid
        })

    except Exception as e:
        print(f"Failed to submit workflow for {protein_name}: {e}")
```

To insert Code Block: Go to Add-ons → Code Blocks
This is a guide: https://www.geeksforgeeks.org/how-to-add-code-block-in-google-docs/

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

Top ten best mutations for 1T5D

Docking_A_310_MET_to_ARG	https://lab	7c84e87e-	2bbc1a03-	Docking	Complete(	22.512083	true	true	auto	[[-19,95,1£	2bbc1a03-	-3.396	-3.396	-3.396
Docking_A_303_ILE_to_THR	https://lab	b3013abc-	26797b94-	Docking	Complete(	24.752563	true	true	auto	[[-19,95,1£	26797b94-	-3.418	-3.418	-3.418
Docking_A_303_ILE_to_LEU	https://lab	da0f6dd8-	96551d1e-	Docking	Complete(	20.206871	true	true	auto	[[-19,95,1£	96551d1e-	-3.438	-3.438	-3.438
Docking_A_303_ILE_to_PRO	https://lab	c5da75f0-	760c4402-	Docking	Complete(	24.901435	true	true	auto	[[-19,95,1£	760c4402-	-3.44	-3.44	-3.44
Docking_A_303_ILE_to_VAL	https://lab	4237c692-	3a8447fd-	Docking	Complete(	24.457315	true	true	auto	[[-19,95,1£	3a8447fd-	-3.444	-3.444	-3.444
Docking_A_303_ILE_to_GLN	https://lab	484e5d24-	bdbbba31-	Docking	Complete(	22.687009	true	true	auto	[[-19,95,1£	bdbbba31-	-3.45	-3.45	-3.45
Docking_A_310_MET_to_LYS	https://lab	a0c07858-	617fac43-	Docking	Complete(	23.102553	true	true	auto	[[-19,95,1£	617fac43-	-3.46	-3.46	-3.46
Docking_A_303_ILE_to_GLU	https://lab	cf435d5f-b	8caef38f-4	Docking	Complete(	22.044594	true	true	auto	[[-19,95,1£	8caef38f-4	-3.466	-3.466	-3.466
Docking_A_303_ILE_to_LYS	https://lab	f70e41d0-	c0f5a7a2-	Docking	Complete(	22.997444	true	true	auto	[[-19,95,1£	c0f5a7a2-	-3.491	-3.491	-3.491
Docking_A_303_ILE_to_GLY	https://lab	e836eb7c-	aa30c775-	Docking	Complete(	23.106946	true	true	auto	[[-19,95,1£	aa30c775-	-3.585	-3.585	-3.585

Top ten mutations for 5BST

	Name	Link	UUID	Protein UL	Item Type	Status	Elapsed Ti	Run Confo	Optimize S	Mode	Pocket	Protein UL	Lowest Do	Lowest Do	Lowest Do
2	Docking_A_243_SER_to_ILE	https://lab	965582fe-	68a76d10-	Docking	Complete(	22.550487	true	true	auto	[[2.266,29	68a76d10-	-5.793	-5.793	-5.793
3	Docking_A_243_SER_to_ASP	https://lab	d0d37528-	dc08bc13-	Docking	Complete(	24.754646	true	true	auto	[[2.266,29	dc08bc13-	-5.781	-5.781	-5.781
4	Docking_A_243_SER_to_GLN	https://lab	79a5d271-	1bd8bbdb-	Docking	Complete(	23.496009	true	true	auto	[[2.266,29	1bd8bbdb-	-5.766	-5.766	-5.766
5	Docking_A_243_SER_to_VAL	https://lab	3319d200-	c25a5fa2-	Docking	Complete(	23.154725	true	true	auto	[[2.266,29	c25a5fa2-	-5.709	-5.709	-5.709
6	Docking_A_306_MET_to_HIS	https://lab	b9a8a710-	8a37200c-	Docking	Complete(	22.455096	true	true	auto	[[2.266,29	8a37200c-	-5.634	-5.634	-4.821
7	Docking_A_344_MET_to_LYS	https://lab	f0df7b0e-	1dd2f3255-	Docking	Complete(	23.583069	true	true	auto	[[2.266,29	dd2f3255-	-5.624	-5.624	-5.624
8	Docking_A_243_SER_to_PHE	https://lab	5182de98-	8f90c0e0-	Docking	Complete(	23.443485	true	true	auto	[[2.266,29	8f90c0e0-	-5.618	-5.618	-5.612
9	Docking_A_243_SER_to_TYR	https://lab	17f772a7-	b3ce4e30-	Docking	Complete(	22.743371	true	true	auto	[[2.266,29	b3ce4e30-	-5.611	-5.611	-4.822
10	Docking_A_344_MET_to_GLN	https://lab	75965c47-	0fe216e1-	Docking	Complete(	23.109794	true	true	auto	[[2.266,29	0fe216e1-	-5.604	-5.604	-5.604
11	Docking_A_243_SER_to_TRP	https://lab	50852f7c-	85d8ca3e-	Docking	Complete(	22.475774	true	true	auto	[[2.266,29	85d8ca3e-	-5.582	-5.582	-4.819

Interpretation

- Looking at the PDBs in pyMOL.
    - For 1T5D:
        - The Met mutations seem to keep the exact position of the ligand. This is super important, so just based on that we should maybe just go for the best Met mutation (to Lys). This is what I propose next.
    - Having more trouble with 5bst. The ligand binding pose changes slightly, but it is only in the non active carboxylic group meaning it is not as important.
        - The best one here seems to be Ser 243 to GLN as it pushes the TPA forward a little bit, making up for the fact that TPA is a bit shorter than 4-coumarate.
- Based on what has been good so far though it seems the Met mutation and the Ile mutation.

Next Steps

- Co-folding for the top mutations and see how it looks.

2025-07-25 Evaluating 1T5D proteinMPNN mutations by docking and co-folding

Contributors: Sixten, Hilya

Objective

Perform and evaluate the best mutations in 1T5D proteinMPNN sequences.

Background

We have managed to get the Rowan API working for automating the point mutations and have done all possible mutations for both proteins. We will take the most relevant mutations and apply them to our proteinMPNN sequences. We found that all the best mutations were involving the residues Ile303 and Met310. We will perform these mutations in pMPNN sequences A to E and evaluate the scores from docking and co-folding.

Important note is that rowan downloads only the ligand with best binding score if the strain is below 4 kcal/mole. Check this in pymol. Important for the table below is that ligand position compared to crystal structure weighed very heavily. This could be dumb, because the active site of the crystal structure and the mutated proteins do not align exactly, meaning that a deviation in ligand position might actually be the correct ligand binding for the mechanism to work. This is however very hard to take into account and will not be discussed further most likely.

Methods and parameters

Component	*Description*
Software Used	Rowan API, with python scripts in vscode
Input Structure	ProteinMPNN sequences A to E for 1T5D.
Key Parameters	Docking score, ligand strain, alignment of ligand position and orientation with crystal structure, pLDDT and ligand affinity (co-folding)
Script Location	API script

Mutations that we want to make: Ile303 and Met310.

The ten best ones from 1T5D:

Docking_A_310_MET_to_ARG	https://lab	7c84e87e-2bbc1a03-	Docking	Complete(	22.512083	true	true	auto	[[-19,95,1£	2bbc1a03-	-3.396	-3.396	-3.396
Docking_A_303_ILE_to_THR	https://lab	b3013abc-26797b94-	Docking	Complete(	24.752563	true	true	auto	[[-19,95,1£	26797b94-	-3.418	-3.418	-3.418
Docking_A_303_ILE_to_LEU	https://lab	da0f6dd8-96551d1e-	Docking	Complete(	20.206871	true	true	auto	[[-19,95,1£	96551d1e-	-3.438	-3.438	-3.438
Docking_A_303_ILE_to_PRO	https://lab	c5da75f0-760c4402-	Docking	Complete(	24.901435	true	true	auto	[[-19,95,1£	760c4402-	-3.44	-3.44	-3.44
Docking_A_303_ILE_to_VAL	https://lab	4237c692-3a8447fd-	Docking	Complete(	24.457315	true	true	auto	[[-19,95,1£	3a8447fd-	-3.444	-3.444	-3.444
Docking_A_303_ILE_to_GLN	https://lab	484e5d24-bdbbba31-	Docking	Complete(	22.687009	true	true	auto	[[-19,95,1£	bdbbba31-	-3.45	-3.45	-3.45
Docking_A_310_MET_to_LYS	https://lab	a0c07858-617fac43-	Docking	Complete(	23.102553	true	true	auto	[[-19,95,1£	617fac43-	-3.46	-3.46	-3.46
Docking_A_303_ILE_to_GLU	https://lab	cf435d5f-8caef38f-4	Docking	Complete(	22.044594	true	true	auto	[[-19,95,1£	8caef38f-4	-3.466	-3.466	-3.466
Docking_A_303_ILE_to_LYS	https://lab	f70e41d0-c0f5a7a2-	Docking	Complete(	22.997444	true	true	auto	[[-19,95,1£	c0f5a7a2-	-3.491	-3.491	-3.491
Docking_A_303_ILE_to_GLY	https://lab	e836eb7c-aa30c775-	Docking	Complete(	23.106946	true	true	auto	[[-19,95,1£	aa30c775-	-3.585	-3.585	-3.585

- Bad ones are:
    - Ile 303 to Gly (Ligand not in pocket)
    - The rest looks good, will look at them in pymol. And send to Hilya for ddG

Top ten mutations for 5BST:

	Name	Link	UUID	Protein UL	Item Type	Status	Elapsed Ti	Run Confo	Optimize S	Mode	Pocket	Protein UL	Lowest Do	Lowest Do	Lowest Do
2	Docking_A_243_SER_to_ILE	https://lab	965582fe-	68a76d10-	Docking	Complete(	22.550487	true	true	auto	[[2.266,29	68a76d10-	-5.793	-5.793	-5.793
3	Docking_A_243_SER_to_ASP	https://lab	d0d37528-	dc08bc13-	Docking	Complete(	24.754646	true	true	auto	[[2.266,29	dc08bc13-	-5.781	-5.781	-5.781
4	Docking_A_243_SER_to_GLN	https://lab	79a5d271-	1bd8bbdb-	Docking	Complete(	23.496009	true	true	auto	[[2.266,29	1bd8bbdb-	-5.766	-5.766	-5.766
5	Docking_A_243_SER_to_VAL	https://lab	3319d200-	c25a5fa2-	Docking	Complete(	23.154725	true	true	auto	[[2.266,29	c25a5fa2-	-5.709	-5.709	-5.709
6	Docking_A_306_MET_to_HIS	https://lab	b9a8a710-	8a37200c-	Docking	Complete(	22.455096	true	true	auto	[[2.266,29	8a37200c-	-5.634	-5.634	-4.821
7	Docking_A_344_MET_to_LYS	https://lab	f0df7b0e-1	dd2f3255-	Docking	Complete(	23.583069	true	true	auto	[[2.266,29	dd2f3255-	-5.624	-5.624	-5.624
8	Docking_A_243_SER_to_PHE	https://lab	5182de98-	8f90c0e0-	Docking	Complete(	23.443485	true	true	auto	[[2.266,29	8f90c0e0-	-5.618	-5.618	-5.612
9	Docking_A_243_SER_to_TYR	https://lab	17f772a7-	b3ce4e30-	Docking	Complete(	22.743371	true	true	auto	[[2.266,29	b3ce4e30-	-5.611	-5.611	-4.822
10	Docking_A_344_MET_to_GLN	https://lab	75965c47-	0fe216e1-	Docking	Complete(	23.109794	true	true	auto	[[2.266,29	0fe216e1-	-5.604	-5.604	-5.604
11	Docking_A_243_SER_to_TRP	https://lab	50852f7c-	85d8ca3e-	Docking	Complete(	22.475774	true	true	auto	[[2.266,29	85d8ca3e-	-5.582	-5.582	-4.819

- Now I have created a folder with all the mutations for Ile 303 and Met 310 (38 x 5). Will do the dockings now in Rowan. A bit worried it will crash the website, but hopefully will work, I mean cmon.
- **pMPNN sequences to use:**
- We are evaluating 5 pMPNN sequences: A to E.
- Will do some cofolding and see how it looks.
- Could not run cofold took to long time (20 minutes, usually takes about 8) so it was cancelled. Not sure why, or how to proceed. (WORKED LATER)

Code

For API script see entry 2025/07/25 (2)

To insert Code Block: Go to Add-ons → Code Blocks
This is a guide: https://www.geeksforgeeks.org/how-to-add-code-block-in-google-docs/

Language: Python

Script:

```
# Python
# This program prints Hello, world!

print('Hello, world!')
```

Results Summary

MPNN sequence	A	B	C	D	E
Mutation	Met 310 → Trp	Met 310 → Ser	Met 310 → Trp	Ile 303 → His	Met 310 → Pro
Docking score	-3,8	-3,7	-4,3	-3,8	-3,3
Strain	1,485	1,584/0,000	3,764	0,000	1,370
Alignment with crystal ligand	Very good	Good	Very good	Very good	Good
Ligand index (important since higher indexes indicates other (often worse) positions has higher binding affinities.	Pos 1	Pos 3	Pos 1	Pos 3	Pos 1
Comment	It was the only mutation for A that was docked correctly. The others ones constantly turn too much and I do not think that is good.	For this one it is important to note that it is pos 3 that matches with the ligand. The other two positions has higher binding affinities but the the position of the ligand is very off. Also, mutant B has significantly higher binding affinities in general, and the Ile 303 → Glu mutation resulted in a binding affinity	Notice high strain here. The docking has other positions with lower strain and very similar binding affinity so this one might be very good actually.	Same thing as for B here.	Looks good. E did however have generally lower docking scores than the others, so maybe not the best to do mutations on.

MPNN sequence	A	B	C	D	E
		of -4,8. However, the position is slightly off (not that much but still a little) so I think this one is not the best.			
Cofold results. pLDDT and binding probability	0,932 and 0,331	0,933 and 0,561	0,933 and 0,356	0,940 and 0,535	0,944 and 0,410

Interpretation

- Looking at the PDBs in pyMOL.
    - For 1T5D:
        - The Met mutations seem to keep the exact position of the ligand. This is super important, so just based on that we should maybe just go for the best Met mutation (to Lys). This is what I propose next.
    - Having more trouble with 5bst. The ligand binding pose changes slightly, but it is only in the non active carboxylic group meaning it is not as important.
        - The best one here seems to be Ser 243 to GLN as it pushes the TPA forward a little bit, making up for the fact that TPA is a bit shorter than 4-coumarate.
- Based on what has been good so far though it seems the Met mutation and the Ile mutation.

Next Steps

- Since docking and to create mutations is almost automatic now I will do the dockings and mutations on the pMPNN sequences as well.
- What needs to be done now:
    - Pick one mutation from each of these based on ddG, docking position and co-fold.
    - Control dock with 4-chlorobenzoate and 4-coumarate
    - Mutate all of the protein MPNN sequences with the one that makes it through.
    - Pick the two best protein MPNN sequences for each of the proteins.

2025-07-27 Docking and co-folding of 5BST proteinMPNN mutated sequences

Contributors: Sixten, Hilya

Objective

Evaluating possible mutations for proteinMPNN sequences and choose the best ones to test for wet lab.

Background

Structurally characterized enzymes, especially with substrate and/or cofactors bound, offer the best opportunities for rational design. However, identifying where mutations should be introduced, is enzyme or enzyme family dependent, as the structural drivers of stereo and regioselectivity will differ.

Reference: Ding, Y., Perez-Ortiz, G., Peate, J., & Barry, S. M. (2022). Redesigning enzymes for biocatalysis: Exploiting structural understanding for improved selectivity. Frontiers in Molecular Biosciences, 9. https://doi.org/10.3389/fmolb.2022.908285

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/) for co-folding and docking
Input Structure	5BST proteinMPNN sequences on the Dry Lab Drive. proteinMPNN 5bst Ligands: 4-coumarate
Key Parameters	Docking score, ligand strain, alignment of ligand position and orientation with crystal structure, pLDDT and ligand affinity (co-folding)
Script Location	API script for 5BST

- Running the mutant D Met310 to Trp docking since there was a weird error for this one and had to be added manually. Double checked and I do not think there were additional errors that needed to be taken into account.
- Downloaded the pdb files for only strain lower than 4 kcal/mol but csv file only showed the lowest score, giving a misleading conclusion. A seems to be fine, but will redo for others. The table above will therefore be the new “best” one.
- Even though the binding affinities are slightly worse the co-folded ligands bind ALOT better (i.e they match super well with the crystal structure ligand). Especially C.
- Cofolding results from only point mutated protein (Met 310 → Lys):

Co-Folding Scores Sequences

Info Notes JSON

Property	Value
Predicted TM-Score (pTM)	0.929
ipTM	0.958
Confidence Score	0.959
Average pLDDT	0.960
Affinity Probability	0.415
Predicted pIC_{50}	3.760
Predicted IC_{50}	173.8 µM

- When evaluating the Met 310 → Lys mutation for the pMPNN sequences it can be said that it has a really good ligand position.

Code

```
import rowan
import time
import json
import shutil
from pathlib import Path
from rowan import smiles_to_stjames, submit_docking_workflow
```

```
# Rowan API key
rowan.api_key = "INSERT API KEY HERE"

# --- Configuration ---
protein_folder =
   Path(r"C:\Users\sixte\OneDrive\Desktop\Skola\Bioinformatics\Docking
   project\proteinMPNN 5bst\Sequences PDB modified coordinates")
ligand_smiles = "[O-]C(=O)c1ccc(C(=O)[O-])cc1"
workflow_base_name = "Docking_"
pocket_box = [[2.266, 29.74, 17.59], [27, 22, 19]]  # Box coordinates

# Convert SMILES once
ligand_molecule = smiles_to_stjames(ligand_smiles)

# === STAGE 1: Submit all workflows ===
submitted_workflows = []

for pdb_file in protein_folder.glob("*.pdb"):
    protein_name = pdb_file.stem
    workflow_name = f"{workflow_base_name}{protein_name}"
    print(f"\n--- Submitting workflow for: {protein_name} ---")

    try:
        protein = rowan.upload_protein(protein_name, pdb_file)
        protein.sanitize()
        time.sleep(5)

        workflow = submit_docking_workflow(
            protein=protein.uuid,
            pocket=pocket_box,
            initial_molecule=ligand_molecule,
            do_csearch=True,
            do_optimization=True,
            name=workflow_name,
        )

        print(f"Submitted: {workflow.uuid}")
        submitted_workflows.append({
            "protein_name": protein_name,
            "pdb_path": str(pdb_file),
            "workflow_uuid": workflow.uuid
```

```
        })

    except Exception as e:
        print(f"❌ Failed to submit workflow for {protein_name}: {e}")
```

Co-folding performed with Rowan GUI.

Results Summary

	A	B	C	D	E
Mutation	Ile 303 → Pro	Ile 303 → Val	Met 310 → Pro	Ile 303 → Leu	Ile 303 → Pro
Docking score	-3,87	-4,0	-4,39	-4,317	-3,38
Strain	1,831	0,946	0,101	2,329	3,416
Alignment with crystal ligand	Good	Ok	Bad	Ok	Ok (but borderline)
Comment	Looks very promising. Position is good, strain is low and docking score is high.	Looks ok. Very similar to the serine mutation done on first screening. Everything is the same. I wonder what the thought process would be here. If all of the Ile mutations are very similar in docking score, strain and position, is there any other way to filter them out?	Position is pretty bad but will see cofold i guesss	Looks ok. Very similar to the serine mutation done on first screening. Everything is the same. I wonder what the thought process would be here. If all of the Ile mutations are very similar in docking score, strain and position, is there any other way to filter them out?	Still same comment as earlier. The energy is consistently lower when compared to the rest, so might not be the best candidate.
Cofold results.	0,937 and 0,379	0,931 and 0,475	0,935 and 0,484	0,943 and 0,453	0,944 and 0,394

	A	B	C	D	E
pLDDT and binding probability					

Interpretation

- The best point mutation seemed to be Met to Lys.

Next Steps

- Look into how the mechanism of acyl-activating enzymes actually work. This is to give an overview of the mechanism of our enzyme and also importantly highlight the difficulties that comes with enzymes that changes conformationally during a mechanism.
- If we want to order the same mutation for the mutated pMPNN sequences as for the single mutations enzyme to compare activity.
- I need to come to a conclusion regarding which of the protein MPNN sequences we should send. There are a couple of things to take into account.
    - Docking score of the mutation
    - Docking score of the mutations relative to mutation on native sequence
    - Position of the mutation compared to ligand.
    - Strain
    - Consistently good across the other sequences (if it is good for all sequences it should have a higher chance of being good).
    - The chosen mutations should be good in different ways and improve the pocket differently. Conclusion here is that we should most likely pick one Met mutation and one Ile mutation

2025-07-28 Docking TPA to unmutated proteinMPNN sequences of 5BST

Contributors: Sixten, Hilya

Objective

Evaluating possible mutations for proteinMPNN sequences and choose the best ones to test for wet lab. We want to see how TPA binds to 5BST to help understand which mutations are favorable, as previously we did not do a control.

Background

Structurally characterized enzymes, especially with substrate and/or cofactors bound, offer the best opportunities for rational design. However, identifying where mutations should be introduced, is enzyme or enzyme family dependent, as the structural drivers of stereo and regioselectivity will differ.

Reference: Ding, Y., Perez-Ortiz, G., Peate, J., & Barry, S. M. (2022). Redesigning enzymes for biocatalysis: Exploiting structural understanding for improved selectivity. Frontiers in Molecular Biosciences, 9. https://doi.org/10.3389/fmolb.2022.908285

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/) for co-folding and docking
Input Structure	5BST proteinMPNN sequences on the Dry Lab Drive. proteinMPNN 5bst Ligands: 4-coumarate
Key Parameters	Docking score, ligand strain, alignment of ligand position and orientation with crystal structure, pLDDT and ligand affinity (co-folding)
Script Location	See previous entry (2025-07-27)

Ran cofolding on native sequence with TPA.

Co-Folding Scores Sequences

Info Notes JSON

Property	Value
Predicted TM-Score (pTM)	0.923
ipTM	0.961
Confidence Score	0.962
Average pLDDT	0.963
Affinity Probability	0.431
Predicted pIC_{50}	3.876
Predicted IC_{50}	133.0 µM

Code

See previous entry (2025-07-27).

Results Summary

Unmutated sequence has actually higher binding affinity with TPA than with Met 310 → Lys mutation.

Interpretation

Conclusion is that it is probably not a very good mutation.

Next Steps

Discuss with Dry Lab team. Why does mutation make affinity worse? How should we proceed? What mutations to test in wet lab?

2025-07-29 Notes: Promising mutations for MPNN sequences?

Contributors: Sixten

Objective

Find the best mutations for MPNN sequences.

Code

```python
import pandas as pd
from collections import defaultdict

# Load Excel file (all sheets as a dictionary of DataFrames)
excel_file = r"EXCELFILE"  # change this to your filename
sheets = pd.read_excel(excel_file, sheet_name=None)

# Track rankings
rankings = defaultdict(list)

# Get the sheet names in a list so we can track index
sheet_names = list(sheets.keys())

for i, sheet_name in enumerate(sheet_names):
    df = sheets[sheet_name]
    top39 = df.head(39)

    for rank, row in top39.iterrows():
        full_obj = row.iloc[1]  # original full object name

        if i == 0:
            # For first sheet: remove everything before the 2nd underscore
            parts = full_obj.split('_', 2)
            obj = parts[2] if len(parts) > 2 else full_obj
        else:
            # For other sheets: remove everything before the 3rd underscore
            parts = full_obj.split('_', 3)
            obj = parts[3] if len(parts) > 3 else full_obj

        rankings[obj].append(rank + 1)
        print(f"{sheet_name}: {obj} at rank {rank + 1}")

# Compute average rank
avg_ranks = {obj: sum(ranks)/len(ranks) for obj, ranks in rankings.items()}

# Sort objects by average rank (lowest is best)
sorted_avg = sorted(avg_ranks.items(), key=lambda x: x[1])

# Print top five results
print("Top 5 objects by average rank:")
for i, (obj, avg_rank) in enumerate(sorted_avg[:5], start=1):
    print(f"{i}. {obj} with average rank {avg_rank:.2f}")
```

Results

- Had a meeting with Edu
- Right now it actually looks like Ile to Ser is the most promising mutation so far since it is consistently good for native enzyme (ok at least) and good across the protein MPNN sequences.
- Will however look into one more. Made a script in python that takes the excel sheets and the mutation with the best average was Met to Pro. Will look at this more. Doing cofolding for all of them.
- Here is the excel, CF Met to Pro.xlsx, with all the cofolding order in best binding probability the highest:

Name	Link	UUID	Protein UL	Item Type	Status	Elapsed Ti	Predicted	ipTM	Confidenc	Average pL	Affinity Prc	Predicted	Predicted
CF_mutant_B_Met_to_Pro	https://lab	d4c7b28f-4	c50a09a1-	Protein–lig	Completed	596.47857	0.919	0.954	0.938	0.934	0.589	0.0000335	4.474
CF_mutant_A_Met_to_Pro	https://lab	3d6e44f8-	cb978853-	Protein–lig	Completed	589.87087	0.924	0.958	0.94	0.936	0.519	0.0000529	4.276
CF_mutant_C_Met_to_Pro	https://lab	f73e44af-8	b63272e5-	Protein–lig	Completed	542.50175	0.915	0.959	0.94	0.935	0.484	0.0001122	3.95
CF_mutant_D_Met_to_Pro	https://lab	5d0ef133-	a6dd2720-	Protein–lig	Completed	556.38657	0.923	0.959	0.94	0.936	0.446	0.0001840	3.735
CF_mutant_E_Met_to_Pro	https://lab	b00d90cf-4	1996f5c2-4	Protein–lig	Completed	584.32607	0.934	0.955	0.946	0.944	0.41	0.0001840	3.735

Next Steps

Try cofolding for proposed mutations.

2025-07-03 Notes on filters for protein mutation workflow

Contributors: Sixten

Objectives

Write down a defined workflow to pick which mutations to test in wet lab.

Background

Even though we have developed several validation workflows using docking, co-folding, and stability evaluations, we can't choose only "the best mutations" to test in wet lab. A general thought is that the mutations we test should all bring something unique to the active site, so that we test different characteristics. This would greatly improve the screening and give hints to which mutations are good and why.

Results

- Made a document where I explain the thought process behind what mutations and sequences are good. Also made summaries in excel, google presentation etc to have everything in a good format. These can obviously be improved.

Link to document: Evalution of sequences

Next Steps

- Will look at some mutations on the native enzyme to see what primers we should order. Based on docking score I will look at:
    - Ile to Lys (Larger and is basic)
    - Ile to Gly (Very small and is aliphatic)
    - Ile to Glu (Larger and is Acidic)
    - Met to Lys (Larger and basic)
    - Ile to Gln (Larger and Amidic)
    - Ile to Pro (Smaller(?) and aliphatic)
- Met to Pro (Smaller(?) and aliphatic)For 5bst we should investigate what mutations are good based on native structure and then go to the protein MPNN sequences with those mutations to decide which of those we want.

2025-07-22 Cofolding of top 10 mutants of 5BST

Contributors: Hilya

Objective

Evaluate proposed point mutations by co-folding.

Background

Sixten has performed and sorted mutations by docking scores and ligand strain. We have decided on 10 mutations to validate by co-folding.

Name	Link	UUID	Protein UL	Item Type	Status	Elapsed Ti	Lowest Do	Lowest Do	Lowest Dc
Docking_A_243_SER_to_ILE	https://lab	965582fe-	68a76d10-	Docking	Completec	22.550487	-5.793	-5.793	-5.793
Docking_A_243_SER_to_ASP	https://lab	d0d37528-	dc08bc13-	Docking	Completec	24.754646	-5.781	-5.781	-5.781
Docking_A_243_SER_to_GLN	https://lab	79a5d271-	1bd8bbdb-	Docking	Completec	23.496009	-5.766	-5.766	-5.766
Docking_A_243_SER_to_VAL	https://lab	3319d200-	c25a5fa2-	Docking	Completec	23.154725	-5.709	-5.709	-5.709
Docking_A_344_MET_to_LYS	https://lab	f0df7b0e-1	dd2f3255-	Docking	Completec	23.583069	-5.624	-5.624	-5.624
Docking_A_243_SER_to_PHE	https://lab	5182de98-	8f90c0e0-	Docking	Completec	23.443485	-5.618	-5.618	-5.612
Docking_A_344_MET_to_GLN	https://lab	75965c47-	0fe216e1-	Docking	Completec	23.109794	-5.604	-5.604	-5.604
Docking_A_306_MET_to_PHE	https://lab	f98a153c-	63594855-	Docking	Completec	23.072356	-5.577	-5.577	-5.577
Docking_A_344_MET_to_GLU	https://lab	ba13c5d4-	f85afa55-8	Docking	Completec	22.682448	-5.554	-5.554	-5.554
Docking_A_243_SER_to_THR	https://lab	3759b522-	b8ab2921-	Docking	Completec	24.13499	-5.556	-5.556	-5.553

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), Google Colaboratory (performing mutations, collect data to a summary table in .csv format)
Input Structure	TPA in smiles format, doubly charged. Given under this section below. 5BST native sequences with the mutations (provided below).
Key Parameters	Predicted TM-Score, ipTM, Confidence Score, Average pLDDT, Affinity Probability, predicted pIC50
Script Location	Read_Cofolding_Output - Colab

TPA, doubly charged, in smiles format
[O-]C(=O)c1ccc(C(=O)[O-])cc1

Original sequence from RCSB PDB
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**S**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**M**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

Residues of interest: 234 Ser, 344 Met

(1)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to ILE**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**I**VLLCGLRVGAAI
LIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**M**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(2)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to ASP**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**D**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**M**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(3)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to GLN**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**Q**VLLCGLRVGA
AILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELED

TVRAKFPNAKLGQGYGMTEAGPVLAMCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNS
LPRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(4)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to VAL**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLNVVLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLAMCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(5)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to LYS**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLNSVLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLAKCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(6)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to PHE**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLNFVLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLAMCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(7)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to GLN**

MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**S**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**Q**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(8)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to GLU**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**S**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**E**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(9)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to PHE**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**S**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**F**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

(10)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 243 SER to THR**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**T**VLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLA**M**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

Code

All co-folding are performed without a script.
Co-folding results collected and written into a table (.csv file) using Google Colaboratory (python).

```
import pandas as pd
import os
from google.colab import drive

# Mount Google Drive with shared drive support
drive.mount('/content/drive', force_remount=True)

# Ask for directory path input
directory_path = input("Enter full path to shared folder: ").strip()

# Get list of CSV files in directory
csv_files = [f for f in os.listdir(directory_path) if
f.endswith('.csv')]

if not csv_files:
    print(f"No CSV files found in: {directory_path}")
    exit()

print(f"Found {len(csv_files)} CSV files to process")

# Initialize combined data list
combined_data = []

# Process each CSV file
for file_name in csv_files:
    file_path = os.path.join(directory_path, file_name)

    try:
        df = pd.read_csv(file_path)
        # Extract filename without extension
        base_name = os.path.splitext(file_name)[0]

        # Create dictionary for row data
        row_data = {'File': base_name}
```

```
        # Add all property-value pairs
        for _, row in df.iterrows():
            property_name = row['Property']
            value = row['Value']
            row_data[property_name] = value

        combined_data.append(row_data)
        print(f"Processed: {file_name}")

    except Exception as e:
        print(f"Error processing {file_name}: {str(e)}")

# Create combined DataFrame
if combined_data:
    combined_df = pd.DataFrame(combined_data)

    # Define output path
    output_path = os.path.join(directory_path, 'combined_results.csv')

    # Save combined results
    combined_df.to_csv(output_path, index=False)
    print(f"\nSuccessfully created combined file: {output_path}")
    print(f"Processed {len(combined_data)}/{len(csv_files)} files")
    print("\nFile preview:")
    print(combined_df.head())
else:
    print("No valid CSV files processed")
```

#README
For shared folders in "Shared with me":
Create a shortcut to the shared folder in your own Drive
Use the path to the shortcut in your Drive (e.g., /content/drive/MyDrive/ShortcutName)
For Shared Drives (Team Drives):
Use the path: /content/drive/Shareddrives/[Shared Drive Name]/[Folder Path]
Example: /content/drive/Shareddrives/Our_Research_Team/Protein_Data
When prompted:
Enter the full path to your shared folder
Make sure you have at least "Viewer" access to the shared folder.

Example of sorting:

```
## this one will sort by Affinity Probability ##

import pandas as pd
import os
from google.colab import drive
```

```
# Mount Google Drive with shared drive support
drive.mount('/content/drive', force_remount=True)

# Ask for directory path input
directory_path = input("Enter full path to shared folder: ").strip()

# Get list of CSV files in directory
csv_files = [f for f in os.listdir(directory_path) if
f.endswith('.csv')]

if not csv_files:
    print(f"No CSV files found in: {directory_path}")
    exit()

print(f"Found {len(csv_files)} CSV files to process")

# Initialize combined data list
combined_data = []

# Process each CSV file
for file_name in csv_files:
    file_path = os.path.join(directory_path, file_name)

    try:
        df = pd.read_csv(file_path)
        base_name = os.path.splitext(file_name)[0]
        row_data = {'File': base_name}

        for _, row in df.iterrows():
            property_name = row['Property']
            value = row['Value']
            row_data[property_name] = value

        combined_data.append(row_data)
        print(f"Processed: {file_name}")

    except Exception as e:
        print(f"Error processing {file_name}: {str(e)}")

# Create combined DataFrame
if combined_data:
    combined_df = pd.DataFrame(combined_data)
```

```
    # Sort by Affinity Probability (lower values first)
    if 'Affinity Probability' in combined_df.columns:
        # Convert to float and sort
        combined_df['Affinity Probability'] = combined_df['Affinity
Probability'].astype(float)
        combined_df = combined_df.sort_values(by='Affinity
Probability', ascending=False)
        print("\nSorted by Affinity Probability (descending)")
    else:
        print("\nWarning: 'Affinity Probability' column not found - no
sorting applied")

    # Define output path
    output_path = os.path.join(directory_path,
'combined_results_sorted_by_AffinityProbability.csv')

    # Save sorted results
    combined_df.to_csv(output_path, index=False)
    print(f"\nSuccessfully created sorted combined file:
{output_path}")
    print(f"Processed {len(combined_data)}/{len(csv_files)} files")

    # Show preview of sorted data
    print("\nSorted data preview (first 5 rows):")
    print(combined_df.head())

    # Show min/max affinity probabilities
    if 'Affinity Probability' in combined_df.columns:
        print(f"\nAffinity Probability range: {combined_df['Affinity
Probability'].min():.3f} to {combined_df['Affinity
Probability'].max():.3f}")
else:
    print("No valid CSV files processed")
```

Example for output:

```
Mounted at /content/drive
Enter full path to shared folder: /content/drive/MyDrive/5BST MPNN Point Mutations
Found 13 CSV files to process
Processed: G_5B_A_pG_5bst.pdb_1st-shell_2nd-shell_treshold-
35_seq_13_A292_MET_to_GLU_cofolding_values.csv
Processed: H_5B_A_pH_5bst.pdb_1st-shell_seq_7_A292_MET_to_GLU_cofolding_values.csv
Processed: G_5B_A_pG_5bst.pdb_1st-shell_2nd-shell_treshold-
35_seq_13_A330_MET_to_LYS_cofolding_values.csv
Processed: H_5B_A_pH_5bst.pdb_1st-shell_seq_7_A330_MET_to_LYS_cofolding_values.csv
Processed: H_5B_A_pH_5bst.pdb_1st-shell_seq_7_A330_MET_to_GLN_cofolding_values.csv
Processed: G_5B_A_pG_5bst.pdb_1st-shell_2nd-shell_treshold-
35_seq_13_A330_MET_to_GLN_cofolding_values.csv
```

```
Processed: I_5B_A_pI_5bst.pdb_1st-shell_treshold-
35_seq_15_A330_MET_to_GLN_cofolding_values.csv
Processed: I_5B_A_pI_5bst.pdb_1st-shell_treshold-
35_seq_15_A330_MET_to_LYS_cofolding_values.csv
Processed: I_5B_A_pI_5bst.pdb_1st-shell_treshold-
35_seq_15_A292_MET_to_GLU_cofolding_values.csv
Processed: J_5B_A_pJ_5bst.pdb_1st-shell_treshold-70_seq_4_A292_MET_to_GLU_cofolding_values.csv
Processed: J_5B_A_pJ_5bst.pdb_1st-shell_treshold-70_seq_4_A330_MET_to_GLN_cofolding_values.csv
Processed: J_5B_A_pJ_5bst.pdb_1st-shell_treshold-70_seq_4_A330_MET_to_LYS_cofolding_values.csv
Error processing combined_results.csv: 'Property'

Sorted by Affinity Probability (descending)

Successfully created sorted combined file: /content/drive/MyDrive/5BST MPNN Point
Mutations/combined_results_sorted_by_AffinityProbability.csv
Processed 12/13 files

Sorted data preview (first 5 rows):
                                                File  \
3   H_5B_A_pH_5bst.pdb_1st-shell_seq_7_A330_MET_to...
2   G_5B_A_pG_5bst.pdb_1st-shell_2nd-shell_treshol...
7   I_5B_A_pI_5bst.pdb_1st-shell_treshold-35_seq_1...
11  J_5B_A_pJ_5bst.pdb_1st-shell_treshold-70_seq_4...
10  J_5B_A_pJ_5bst.pdb_1st-shell_treshold-70_seq_4...

    Predicted TM-Score (pTM)   ipTM  Confidence Score  Average pLDDT  \
3                      0.930  0.960             0.933          0.926
2                      0.923  0.960             0.964          0.965
7                      0.912  0.961             0.963          0.964
11                     0.880  0.957             0.930          0.923
10                     0.896  0.949             0.938          0.935

    Affinity Probability  Predicted pIC50  Predicted IC50
3                  0.459            4.730        0.000019
2                  0.424            4.303        0.000050
7                  0.363            3.850        0.000141
11                 0.355            3.871        0.000135
10                 0.293            3.370        0.000427

Affinity Probability range: 0.193 to 0.459
```

Results Summary

- Collected the cofolding summary results in this file (csv): [combined_results.csv]
- Collected the PDB files in this folder: [5BST Point Mutations - Google Drive]
- From the results the best mutations are 5, 7, 8, 9, 3 according to both affinity and pIC50
    - However better affinity than native is only achieved by 5, 7, 8
    - Better pIC50 than native is only achieved by 5
- Checked these 5 structures in pyMOL, aligned with crystal structure of 5BST

A quick look, nothing too wonky in the protein structure of all 5 mutations. The cofolding seems correct. All substrates do bind in the correct pocket, they're all kinda relatively good, but the best seem to be 5,7,8. These three have the same orientation of the benzene ring and the planes of the benzene rings between the native substrate and TPA overlap a lot.

Interpretation

From the results, the best mutations are 5, 7, 8, 9, 3 according to both affinity and pIC50.

Next Steps

Choose the best mutations to perform in wet lab. We are going to order the original sequences and do site-directed mutagenesis, so we need to design the primers.

2025-07-22 Cofolding and ddG evaluation of top 10 mutants of 5BST (corrections) and some proteinMPNN sequences

Contributors: Hilya

Objective

Evaluate proposed point mutations by co-folding (see entry 2025-07-31).
Native 5BST: top 10 mutations sorted by docking result.
ProteinMPNN sequences: A to J, except for F

Background

See entry 2025-07-31. There are some mistakes in the previous experiments on mutation 8 that it should have been 306 Met instead of 344 Met. We will redo it.

Methods and parameters

Component	*Description*
Software Used	Rowan (https://labs.rowansci.com/), Google Colaboratory (performing mutations, collect data to a summary table in .csv format)
Input Structure	TPA in smiles format, doubly charged. Given under this section below. Sequences given under this section below. 5BST native sequences with the mutations (provided below). ProteinMPNN sequences: proteinMPNN 5bst
Key Parameters	Predicted TM-Score, ipTM, Confidence Score, Average pLDDT, Affinity Probability, predicted pIC50
Script Location	Read_Cofolding_Output - Colab

Sequences used this time for the native 5BST:

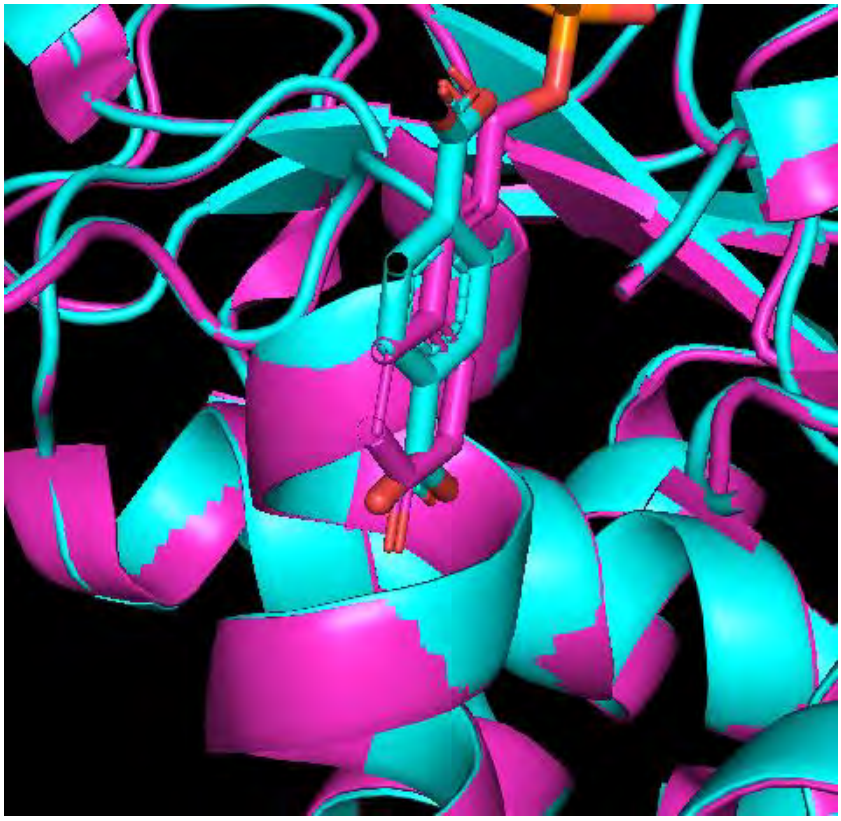
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV

TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLNSVLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVMSGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLAMCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

Residues of interest: 234 Ser, 306 Met, 344 Met

(8)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to GLU**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGIQPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIV
TQACHVNKVKDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGT
TGLPKGVMLTHKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLNSVLLCGLRVGAA
ILIMQKFDIVSFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTVESGAAPLGKELEDT
VRAKFPNAKLGQGYGMTEAGPVLAMCLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSL
PRNQSGEICIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYK
GFQVAPAELEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIF
YKRIKRVFFVDAIPKSPSGKILRKDLRAKLAAGLPN

Re-running them in Rowan cofolding.



Correct pocket and good alignment like mutation number 5, 7, and the other (wrong) 8 (344Met to Glu). Here 5bst is the pink one.

- Ran FoldX for 10+1 mutations that I co-folded (10 from Sixten and 1 wrong one). Uploaded the results to the folder.

(8)
>5BST_1|Chain A|4-coumarate--CoA ligase 2|Nicotiana tabacum (4097)|**Mutated 344 MET to PHE**
MEKDTKQVDIIFRSKLPDIYIPNHLPLHSYCFENISEFSSRPCLINGANKQIYTYADVELNSRK
VAAGLHKQGI
QPKDTIMILLPNSPEFVFAFIGASYLGAISTMANPLFTPAEVVKQAKASSAKIIVTQACHVNKV
KDYAFENDVKIICIDSAPEGCLHFSVLTQANEHDIPEVEIQPDDVVALPYSSGTTGLPKGVML
THKGLVTSVAQQVDGENPNLYIHSEDVMLCVLPLFHIYSLN**S**VLLCGLRVGAAILIMQKFDIV
SFLELIQRYKVTIGPFVPPIVLAIAKSPMVDDYDLSSVRTV**F**SGAAPLGKELEDTVRAKFPNA
KLGQGYGMTEAGPVLA**M**CLAFAKEPFEIKSGACGTVVRNAEMKIVDPKTGNSLPRNQSGEI
CIRGDQIMKGYLNDPEATARTIDKEGWLYTGDIGYIDDDDELFIVDRLKELIKYKGFQVAPAE
LEALLLNHPNISDAAVVPMKDEQAGEVPVAFVVRSNGSTITEDEVKDFISKQVIFYKRIKRVF
FVDAIPKSPSGKILRKDLRAKLAAGLPN

Code

See entry 2025-07-31.
Language: Python

Results Summary

- Uploaded results on 5BST Point Mutations:
- Updates all csv files to contain the correct mutations.
- Uploaded energies.
- J performs poorly for pIC50.
- Energies do not reveal very much information, most mutations results in zero dGG.

Interpretation

For native 5BST: both pIC50 and the affinity the best mutation is H 330Met to Lys.
The best mutations seem to be Met344 to Lys and Met 344 to Gln. The best pMPNN sequences for 5BST seem to be H and J, based on both cofolding and dockings.
Maybe the changes in ddG are too small to be picked up by FoldX, or the algorithm was stuck in a local minima.

Next Steps

Choose the best mutations to perform in wet lab. We are going to order the original sequences and do site-directed mutagenesis, so we need to design the primers.