Algorithms in Structural Bioinformatics

Assignment #2

Konstantinos Giatras

Exercise 1

The 3D structure of the human serotonin receptor was determined in the absence and in the presence of serotonin (PDB id: 7e2x, 7e2y). For this exercise, we will be going through the information/figures provided in PDBe regarding the 7e2y entry (https://www.ebi.ac.uk/pdbe/entry/pdb/7e2y).

Question 1.1

To find the **method** by which the 3D structure of 7e2y was determined, we download the pdb file and locate the appropriate information in the file. In this case, the structure was determined using **cryo-EM** experiments. This is further confirmed by the relevant paper (https://www.nature.com/articles/s41586-021-03376-8).
The **resolution** of the structure can be found either on the PDBe entry or the downloaded pdb file. For 7e2y, the resolution is **3.00 Angstroms**.

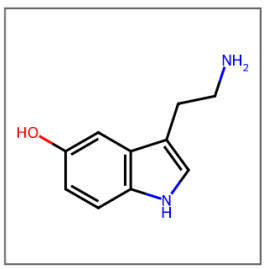
Question 1.2

The **number of chains and amino acids in each chain** can be found under the structure analysis section of the 7e2y PDBe entry. They are as follows:

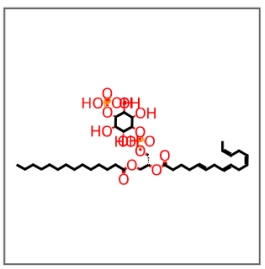
No	Chain	Amino Acids
1	A	354
2	B	345
3	G	71
4	R	543

Question 1.3

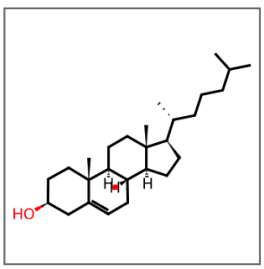
The **ligands** present in the structure can be found under the ligands and environments section of the 7e2y PDBe entry. These include 1x **serotonin** (SRO), 4x **cholesterols** (CLR) and 1x [(2R)-1-[oxidanyl-[(2R,3R,5S,6R)-2,3,5,6-tetrakis(oxidanyl)-4-phosphonooxy-cyclohexyl]oxy-phosphoryl]oxy-3-tetradecanoyloxy-propan-2-yl] (5E,8E)-hexadeca-5,8,11,14-tetraenoate (**J40**) molecule.



1 x SRO



1 x J40



4 x CLR

Exercise 2

For this exercise, we will be using Chimera-X (https://www.cgl.ucsf.edu/chimerax/), a molecular visualization program, to prepare different figures of the overall 7e2y structure of the complex (this website: https://www.cgl.ucsf.edu/chimerax/docs/user/commands/colortables.html, was used to learn the appropriate coloring commands).

Question 2.1

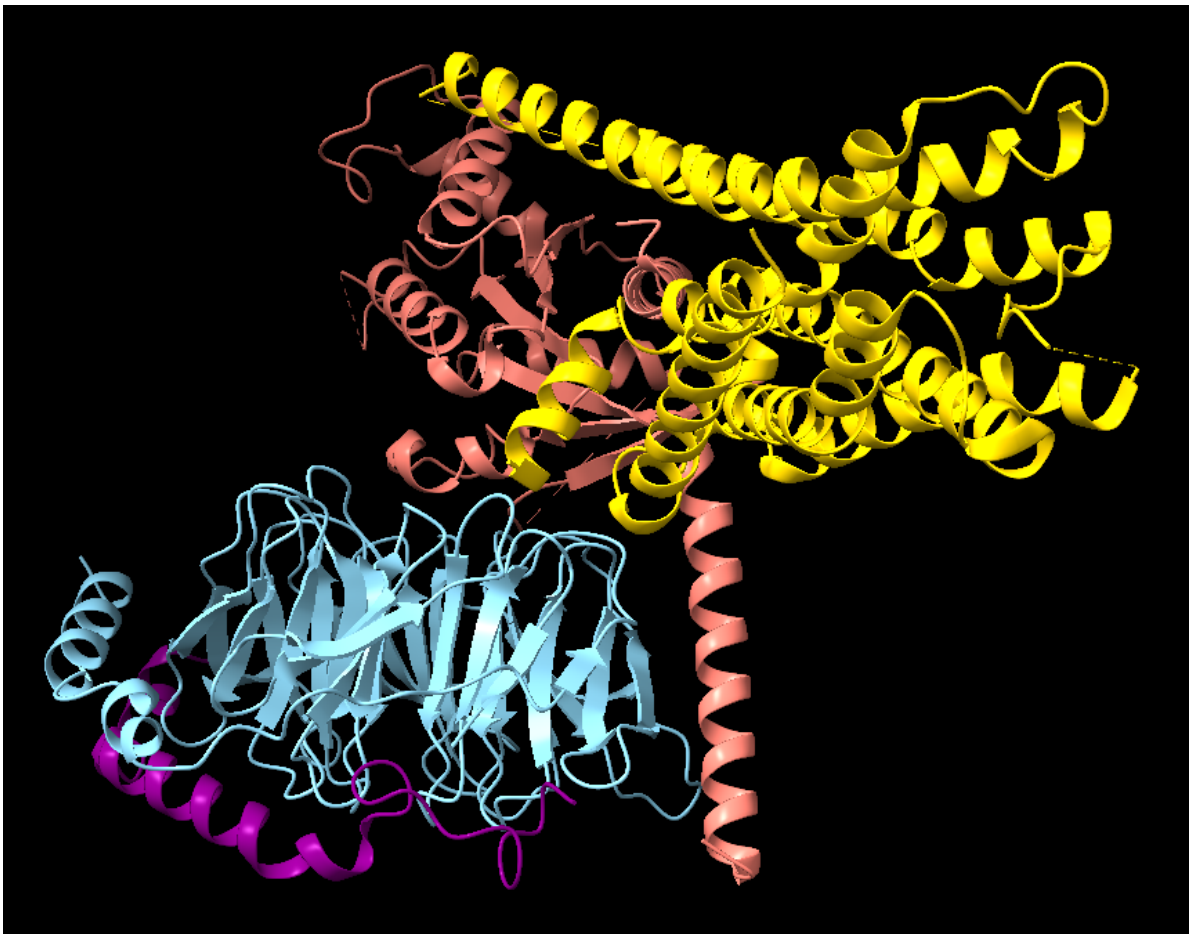


Figure 1: Secondary Structure Elements of the Protein Coloured by Chain (A: salmon, B: sky blue, G: purple, R: gold)

Question 2.2

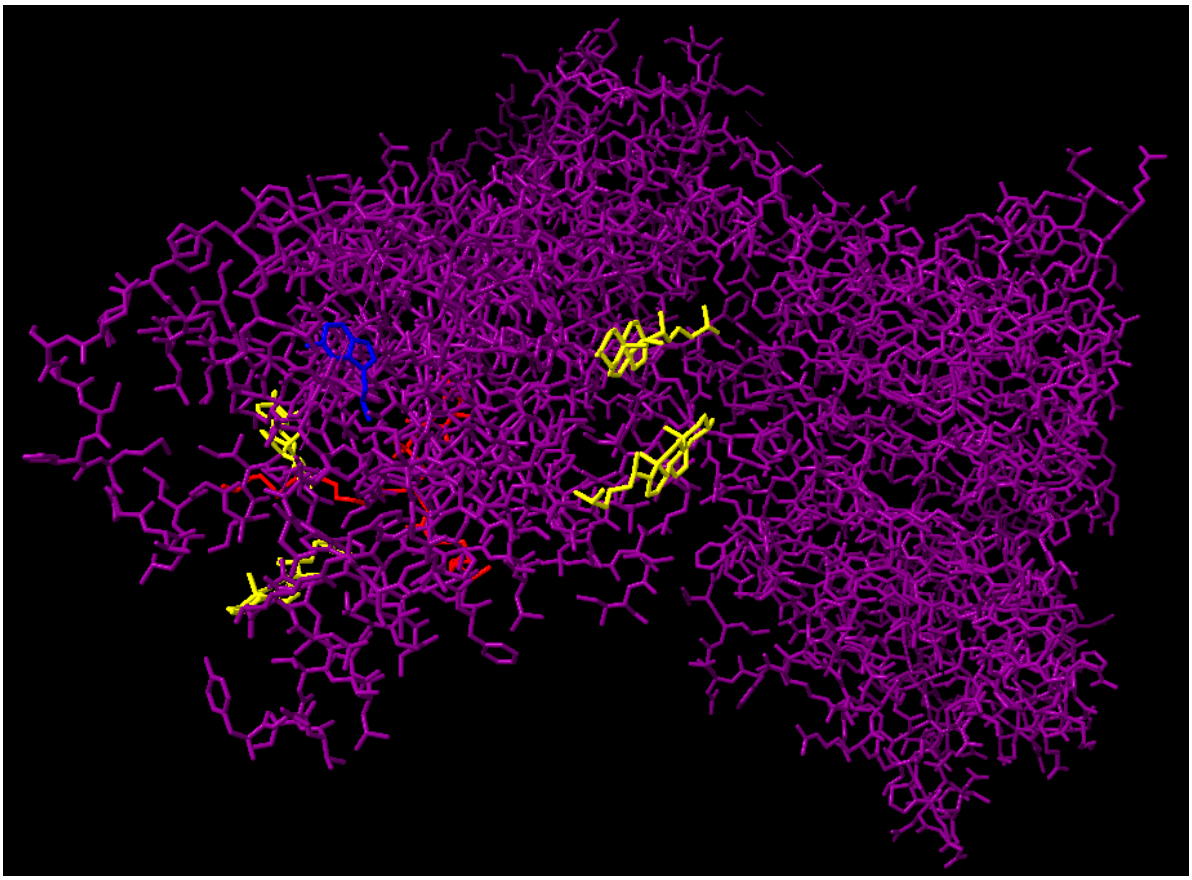


Figure 2: Chemically Distinct Molecules in the Structure (Human Serotonin Receptor: purple, Serotonin: blue, Cholesterol: yellow, [(2R)-1-[oxidanyl-[(2R,3R,5S,6R)-2,3,5,6-tetrakis(oxidanyl)-4-phosphonooxy-cyclohexyl]oxy-phosphoryl]oxy-3-tetradecanoyloxy-propan-2-yl] (5E,8E)-hexadeca-5,8,11,14-tetraenoate: red)

Exercise 3

For this exercise (based on this article: https://www.ebi.ac.uk/pdbe/news/serotonin-ruffling-feathers), we will determine the c-RMSD between the human (7e2y, deposited in PDBe, determined experimentally) and chicken (D2K8P9, predicted using AlphaFold, https://alphafold.ebi.ac.uk/entry/D2K8P9) serotonin 1A receptor structures (5-HT1A), mainly using commands on the console of the PyMOL software, which provides a Python API for structural analysis.

Question 3.1

To determine the RMSD over all atoms, we begin by loading the two pdb files into PyMOL. We then type the following commands:

```
align 7e2y, AF-D2K8P9-F1-model_v4, cycles=0, object=aln
rms_cur 7e2y & aln, AF-D2K8P9-F1-mod & aln, matchmaker=-1
```

The *'align'* command (https://pymolwiki.org/index.php/Align) performs a sequence alignment followed by a structural superposition, and then carries out zero or more cycles of refinement in order to reject structural outliers found during the fit. The all-atom RMSD can be obtained by setting *'cycles=0'* and thus not doing any outlier rejection. The *'object'* argument creates and names an alignment object to be used by the next command.
The *'rms_cur'* command (https://pymolwiki.org/index.php/Rms_cur) computes the RMS difference between two atom selections without performing any fitting. By default, only matching atoms in both selections are used for the fit. The previously generated alignment object is used in the atom selection and the *'matchmaker=-1'* argument (for matching atom pairs) is used to assume that atoms are stored in the identical order.

The output in the console is: **all-atom RMSD = 9.131 Angstrom (2162 to 2162 atoms)**, indicating a relatively high level of structural deviation between the two proteins, with significant differences in the positions of all atoms. The comparison is subject to possible errors, since it was conducted between predicted and experimentally determined structures.

Question 3.2

To determine the RMSD over C_α atoms, we begin by loading the two pdb files into PyMOL. We then type the following commands:

```
align 7e2y, AF-D2K8P9-F1-model_v4, object=aln
rms_cur 7e2y & aln and name ca, AF-D2K8P9-F1-mod & aln and name ca, matchmaker
    =-1
```

In the *'align'* command, we remove the *'cycles'* argument and in the *'rms_cur'* command, we add two *"and name ca"* arguments to calculate the RMSD only for the C_α atoms.

The output in the console is: **C_α RMSD = 1.100 Angstrom (235 to 235 atoms)**, indicating a relatively higher degree of similarity in the backbone conformation of the two proteins. However, there may still be variations in the side chain orientations and overall protein structure.

Question 3.3

To determine the RMSD using the secondary structure elements only, we need to define the secondary structure elements (SSEs) and align the corresponding regions between the two structures. One way to do this is by using the DSSP (Define Secondary Structure of Proteins) online tool (http://bioinformatica.isa.cnr.it/SUSAN/NAR2/dsspweb.html), an application that assigns secondary structure to proteins, and a script in Visual Studio Code utilizing the biopython library in tandem. After generating the dssp files from their corresponding pdb files, we run the following code:

```
from Bio.PDB import PDBParser, Superimposer

# PDB file paths
pdb1_path = "7e2y.pdb"
pdb2_path = "AF-D2K8P9-F1-model_v4.pdb"

# DSSP file paths
dssp1_path = "7e2y.dssp"
dssp2_path = "AF-D2K8P9-F1-model_v4.dssp"

# Parse the PDB files to get the structures
pdb_parser = PDBParser()
structure1 = pdb_parser.get_structure("7e2y", pdb1_path)
structure2 = pdb_parser.get_structure("AF-D2K8P9-F1-model_v4", pdb2_path)

# Parse the DSSP files to get the secondary structure elements
def parse_dssp(dssp_path):
    with open(dssp_path) as f:
        lines = f.readlines()[28:]  # DSSP secondary structure starts from line 29
        return [line[16] for line in lines if len(line) > 16 and line[16] in 'EHBIGST']

dssp1 = parse_dssp(dssp1_path)
dssp2 = parse_dssp(dssp2_path)

# Get the atom coordinates for secondary structure elements
def get_secondary_structure_atoms(structure, dssp):
    atoms = []
    for model in structure:
        for chain in model:
            for i, residue in enumerate(chain):
                # Get the residue number
                res_num = residue.get_id()[1]

                # If we have a secondary structure element for this residue
                if res_num < len(dssp) and dssp[res_num] != '-':
                    try:
                        atoms.append(residue["CA"])
                    except:
```

```
                        continue
    return atoms

atoms1 = get_secondary_structure_atoms(structure1, dssp1)
atoms2 = get_secondary_structure_atoms(structure2, dssp2)

# Modify this section to use the smaller number of atoms from atoms1 and
    atoms2
min_length = min(len(atoms1), len(atoms2))
super_imposer = Superimposer()
super_imposer.set_atoms(atoms1[:min_length], atoms2[:min_length])

print("secondary structure elements RMSD:", super_imposer.rms, "Angstrom")
```

In the beginning, the script imports necessary modules and defines file paths for the pdb and the dssp files. The pdb files contain the 3D atomic coordinates of the proteins, while the dssp files contain information about the secondary structure elements (alpha helices, beta sheets, etc.) of the proteins. The PDBParser object is used to parse the pdb files and get the protein structures. Next, the dssp files are parsed using the *'parse_dssp'* function to get the secondary structure elements. This function reads each line of the dssp file starting from line 29 (as the actual dssp data begins from this line) and stores the secondary structure code (at position 16 in each line) if it corresponds to a known secondary structure element.
Subsequently, the *'get_secondary_structure_atoms'* function is used to extract only the alpha carbon ('CA') atoms corresponding to secondary structure elements from each protein structure. It does so by iterating through each model, chain, and residue of the structure, and checking if the residue number corresponds to a secondary structure element in the dssp data.
Finally, the script calculates and prints the RMSD of the secondary structure elements between the two structures using the *'Superimposer'* class from BioPython. The RMSD is computed only for the smaller number of atoms between the two structures. The RMSD value gives a quantitative measure of the similarity between the secondary structure elements of the two protein structures.

The output in the VSCode terminal is: **secondary structure elements RMSD: 34.215 Angstrom**, which suggests significant conformational differences in the secondary structures, but this doesn't negate the potential for similar functionality. The comparison is subject to possible errors and uncertainties, since it was conducted between predicted and experimentally determined structures whose secondary structure elements do not align properly.